À tradire
ISSN : 2968-3912

1 | 2022
L'apprenant de et par la traduction

Le fonctionnement de la traduction automatique neuronale

Juan Antonio Pérez-Ortiz, Mikel L. Forcada Felipe Sánchez-Martínez

Sébastien Palmieri Romain Revet

https://lodelpreprod.univ-rennes2.fr/atradire/index.php?id=187

DOI : 10.56078/atradire.187

Juan Antonio Pérez-Ortiz, Mikel L. Forcada Felipe Sánchez-Martínez, « Le fonctionnement de la traduction automatique neuronale », *À tradire* [], 1 | 2022, 20 décembre 2022, 28 septembre 2026. URL : https://lodelpreprod.univ-rennes2.fr/atradire/index.php?id=187

Licence Creative Commons – Attribution 4.0 International – CC BY 4.0

Le fonctionnement de la traduction automatique neuronale

Juan Antonio Pérez-Ortiz, Mikel L. Forcada Felipe Sánchez-Martínez

Sébastien Palmieri Romain Revet

---

1. Introduction
2. Une analogie imparfaite entre traduction humaine et TAN
3. Les réseaux de neurones artificiels
   3.1. Neurones artificiels
   3.2. Neurones et réseaux
   3.3. Couches de neurones
   3.4. La traduction automatique neuronale
   3.5. L'entraînement des réseaux de neurones
   3.6. La généralisation dans les réseaux de neurones
4. Le plongement lexical : une représentation vectorielle des mots
   4.1. La généralisation
   4.2. Les propriétés géométriques des plongements lexicaux
5. Les plongements lexicaux contextuels et l'attention
   5.1. Plusieurs couches d'attention valent mieux qu'une
   5.2. Plusieurs têtes valent mieux qu'une
   5.3. Les plongements lexicaux contextuels et le traitement automatique du langage naturel
6. Et enfin, la traduction automatique neuronale
   6.1. Le modèle Transformer : un encodeur-décodeur basé sur l'attention
   6.2. Architecture récurrente
7. Paramètres supplémentaires
   7.1. Mots et sous-mots
   7.2. Critères d'arrêt et d'évaluation
   7.3. Recherche en faisceau

Conclusions

Le présent document est la traduction par Sébastien Palmieri et Romain Revet d'une version préimpression du chapitre paru dans le manuel *Machine translation for everyone: Empowering users in the age of artificial intelligence*, l'un des résultats du projet https://multitrainmt.eu cofinancé par le programme Erasmus de l'Union européenne. Ce livre a été publié sous la direction de Dorothy Kenny en 2022 et est en libre accès à l'adresse https://langsci-press.org/catalog/book/342. Le chapitre 7 dont nous vous proposons une traduction est accessible, dans sa version originale anglaise, à cette adresse : https://zenodo.org/record/6760020/files/342-Kenny-2022-8.pdf?download=1.

1. Introduction

1 Pour commencer, il faut savoir que la traduction automatique neuronale (TAN) traite la traduction comme une tâche impliquant des opérations sur des nombres, effectuées par des systèmes mathématiques appelés *réseaux de neurones artificiels*. En d'autres termes, les systèmes prennent une phrase et la transforment en une série de nombres. Ils les additionnent ensuite avec d'autres nombres (généralement plusieurs milliers, voire des millions), les multiplient par d'autres nombres, effectuent quelques opérations mathématiques supplémentaires relativement simples, et génèrent finalement une traduction de la phrase originale dans une autre langue.

2 Peut-être avez-vous toujours envisagé la traduction sous un angle différent : telle une opération intellectuelle qui implique des processus cognitifs difficiles à expliciter, et se déroulant dans certaines zones profondes du cerveau humain. Et vous auriez tout à fait raison ! Cependant, l'approximation actuellement réalisée par les ordinateurs suit un tout autre chemin : des millions d'opérations mathématiques sont effectuées en une fraction de seconde pour aboutir à une traduction qui peut parfois être qualifiée de correcte, mais pas toujours. Or, il s'avère que la proportion de traductions jugées correctes a augmenté de façon spectaculaire ces dernières années. D'un point de vue historique, le fonctionnement des *réseaux de neurones artificiels* a pourtant été élaboré sur la base d'un modèle simplifié de *réseau de neurones naturels* semblable à celui qui constitue notre cerveau. En effet, les processus cognitifs qui s'y déroulent sont également le produit de calculs neuronaux distribués similaires aux opérations mathématiques mentionnées ci-dessus.

3 Ce chapitre va vous présenter les points clefs de la technologie de TAN. Nous commencerons par faire le lien entre la manière dont la traduction s'effectue dans le cerveau humain et la procédure utilisée par un système TAN. Cette étape va nous permettre d'introduire les notions de base nécessaires à une compréhension approfondie des principes de l'*apprentissage automatique* et des *réseaux de*

neurones artificiels, qui constituent deux des pierres angulaires de la TAN. Ensuite, nous aborderons le concept fondamental de *plongement lexical non contextuel*. Une représentation informatisée des mots dotés de diverses propriétés intéressantes qui, lorsqu'elles sont combinées par un procédé nommé *attention*, produisent le *plongement lexical contextuel*. Ce dernier étant indispensable à l'accomplissement de la TAN. Tous ces éléments nous permettront de brosser un tableau du fonctionnement interne des deux modèles de TAN les plus utilisés, à savoir le modèle *Transformeur* et le modèle *récurrent*. Pour finir, nous vous présenterons une série de notions annexes qui vous permettront d'améliorer vos connaissances sur ce qui se produit en coulisse de ces systèmes.

2. Une analogie imparfaite entre traduction humaine et TAN

4 Pour simplifier un peu les choses, faisons l'hypothèse rapide que la traduction d'un texte revient à traduire chacune de ses phrases indépendamment les unes des autres. Supposons maintenant un instant que la traduction d'une phrase est un processus en deux étapes : le traducteur détermine d'abord l'*interprétation* ou le *sens* de l'intégralité de la phrase source, puis rédige d'une seule traite, dans la langue cible, une phrase à l'interprétation similaire. Or, les traducteurs sont confrontés tous les jours à des phrases jamais rencontrées auparavant telles que « Le crayon m'a glissé des mains, s'est levé et a commencé à me parler. », et parviennent tout de même à les traduire. Comment est-ce possible ? La linguistique a apporté une réponse à cette question sous la forme d'un principe, le *principe de compositionnalité sémantique* : nous *construisons* l'interprétation de chaque phrase en combinant les interprétations individuelles des mots qui la composent. De plus, l'ordre dans lequel ces interprétations sont combinées est dicté par la structure syntaxique de la phrase : les mots forment des groupes de mots, ces groupes de mots se combinent pour former des groupes de mots plus longs, et ce jusqu'à l'obtention de l'intégralité de la phrase. Un traducteur analyserait ensuite cette interprétation et effectuerait la démarche inverse, mais dans la langue cible. Bien sûr, les traducteurs ne traitent pas les phrases dans leur ensemble, surtout lorsqu'elles sont longues. Des

raccourcis peuvent être empruntés pour éviter l'élaboration d'interprétations de phrases entières, mais tenons-nous-en à cette version simplifiée pour le moment.

5 La TAN fonctionne de manière analogue lors de la traduction d'une phrase. Pendant la phase d'*encodage*, le système attribue une *représentation* neuronale individuelle, ou *plongement*, à chaque mot du texte source. Ces représentations neuronales sont ensuite combinées pour générer une représentation similaire, mais cette fois au niveau phrastique. Lors de cette étape, ces représentations individuelles sont également modifiées en fonction de leur contexte ; on pourrait donc considérer qu'il s'agit d'une représentation contextualisée de l'interprétation ou du sens. Ensuite, durant la phase de *décodage*, les représentations phrastiques sont progressivement décomposées pour prédire un à un les mots de la phrase cible. L'*encodeur* et le *décodeur* qui effectuent ces deux phases sont en fait des *réseaux de neurones artificiels* interconnectés en un seul réseau de neurones hybride.

6 À l'instar de ce que font les traducteurs, le fonctionnement des architectures neuronales actuelles ne repose pas véritablement sur l'examen de l'intégralité de la phrase source lors de la production de chaque mot cible. Ces architectures ont plutôt appris à *porter leur attention* sur les mots sources pertinents et les mots cibles déjà produits.

7 Dans les sections suivantes de ce chapitre, nous ferons une description détaillée de la nature de ces *représentations*, de la structure des *réseaux de neurones artificiels* (que nous appellerons désormais simplement *réseaux de neurones*) qui les construisent et les modifient en prêtant une *attention* sélective aux éléments importants, ainsi qu'aux façons dont ces réseaux de neurones artificiels peuvent être *entraînés* à effectuer cette tâche à l'aide d'exemples de traduction.

3. Les réseaux de neurones artificiels

8 Pour comprendre ce qu'est la TAN, il faut se pencher de plus près sur les réseaux de neurones artificiels (Goodfellow *et al.* 2016) qui l'effec-

tuent : de quoi sont-ils constitués ? Comment fonctionnent-ils et comment sont-ils entraînés ?

9 Le mot *neurone* évoque clairement le fonctionnement du système nerveux des animaux, et en particulier celui du cerveau humain. Les réseaux de neurones artificiels sont en effet constitués de milliers voire de millions d'unités artificielles semblables à des neurones dont l'*activation* (c'est-à-dire le degré d'*excitation* ou d'*inhibition*) dépend des signaux qu'ils reçoivent des autres neurones et de la force des connexions les transmettant.

Schéma 1



Mise à jour de l'état *S4* du neurone artificiel 4 en réponse aux stimuli reçus des neurones 1, 2 et 3

3.1. Neurones artificiels

10 Les neurones artificiels sont les principaux éléments qui entrent dans la constitution des réseaux de neurones artificiels. Lors de la mise à jour de leur état, ou activation, le fonctionnement de ces neurones artificiels (que nous appellerons simplement *neurones* à partir de maintenant) peut être décrit comme un processus en deux étapes. Imaginons la situation simplifiée du schéma 1 dans laquelle nous étudions comment l'activation du neurone S_4 est mise à jour en réponse aux stimuli reçus des neurones S_1, S_2 et S_3.

11 Dans un premier temps, les activations des neurones S_1, S_2 et S_3, tous connectés au neurone S_4, sont additionnées après avoir été multipliées séparément par un poids (w_1, w_2 et w_3) représentant la force de leurs connexions. Ces poids, positifs ou négatifs, déterminent la façon dont leurs activations sont converties en stimuli concrets pour le neurone S_4. Par exemple, si le poids w_2 est positif et que l'activation de S_2 est élevée, cela contribuera à exciter le neurone S_4 (stimulus positif) ; en revanche, un poids w_2 négatif entraînera l'inhibition du neurone S_4 (stimulus négatif). De manière générale, les neurones connectés par des poids positifs ont tendance à être simultanément excités ou inhibés, tandis que les neurones connectés par des poids

négatifs tendent à être dans les états inverses. Pour en revenir au neurone S_4, si on additionne les stimuli provenant de chaque neurone, on obtient un stimulus net :

$$x = w_1 \text{ x } S_1 + w_2 \text{ x } S_2 + w_3 \text{ x } S_3 \text{ (7.1)}$$

12 Le stimulus net *x* peut adopter n'importe quelle valeur possible, positive ou négative, mais ne marque pas encore l'activation du neurone S_4. En effet, c'est dans un deuxième temps que le neurone S_4 *réagit* à ce stimulus. Dans l'exemple précédent, lorsque le stimulus est de valeur intermédiaire, c'est-à-dire ni trop positive ou ni trop négative, le neurone S_4 y est très sensible. Cependant, lorsque les stimuli sont très négatifs ou très positifs, leur valeur réelle importe peu, car le neurone est respectivement fortement inhibé ou fortement excité.

Schéma 2



Réaction d'un neurone au stimulus total reçu

13 Dans notre exemple, le neurone S_4 est tel que son activation est comprise entre -1 et +1. Le schéma 2 représente la réaction du neurone S_4 au stimulus de l'équation 7.1. Cette réaction est représentée par une fonction *F* (...), appelée *fonction d'activation*, qui est appliquée au stimulus et dont le résultat équivaut à l'activation de S_4 :

$$S_4 = F(x) = F\ (w_1 \text{ x } S_1 + w_2 \text{ x } S_2 + w_3 \text{ x } S_3). \text{ (7.2)}$$

14 Comme vous pouvez le constater, pour des valeurs proches de 0 sur les axes horizontaux, la réaction est proportionnelle au stimulus. En revanche, lorsque les stimuli sont fortement positifs ou négatifs, et que le neurone est très inhibé ou très excité, la réaction est beaucoup plus faible. Pour ce genre de neurones, les valeurs extrêmes réelles de -1 et +1 ne sont jamais atteintes, quelle que soit la force totale du stimulus. Comme nous l'avons dit précédemment, le neurone S_4 de notre exemple appartient à une catégorie de neurone spécifique dont l'activation varie entre -1 et +1. Il existe d'autres types de fonctions

d'activation d'amplitudes variables, mais nous ne les présenterons pas dans ce chapitre.

3.2. Neurones et réseaux

15 Des neurones tels que celui évoqué dans la section précédente peuvent être connectés pour former un réseau de neurones artificiels exécutant une tâche de calcul spécifique pour résoudre un problème donné. Au sein d'un réseau, certains neurones reçoivent des stimuli extérieurs qui servent de *données d'entrée* au réseau de neurones (tout comme nos yeux sont connectés à notre cerveau et lui transmettent des images) et représentent un problème à résoudre. Un autre type de neurone reçoit uniquement des stimuli en provenance d'autres neurones (*neurones cachés*), et d'autres neurones enfin, nommés *neurones de sortie*, représentent la solution au problème (un peu comme les signaux envoyés aux muscles d'une de vos mains pour qu'ils bougent de manière spécifique).

Schéma 3



Un réseau de neurones artificiels avec trois neurones cachés et deux neurones de sortie. Chaque connexion a un poids qui n'est pas indiqué sur le schéma.

16 Le schéma 3 nous montre l'exemple du fonctionnement de cinq neurones au sein d'un réseau de neurones. Ce réseau reçoit trois données d'entrées, qui sont transmises à trois neurones cachés qui, à leur tour, stimulent deux neurones de sortie.

17 Lorsqu'on construit un réseau de neurones pour résoudre un problème spécifique, il faut d'abord définir son *architecture* : combien de neurones possède-t-il ? Comment sont-ils connectés ? Quels neurones reçoivent les données d'entrées extérieures au réseau et quels neurones sont désignés comme neurones de sortie ? Cependant, le calcul effectué dépend avant tout des poids de toutes les connexions au sein du réseau. Une caractéristique intéressante des réseaux de neurones artificiels est qu'ils peuvent être *entrainés* à effectuer une tâche à partir d'exemples, c'est-à-dire que leurs poids peuvent être fixés à des valeurs spécifiques en observant un

ensemble de situations déjà rencontrées. La valeur de chaque poids étant composée des valeurs des signaux d'entrée représentant les problèmes et des valeurs des activations de sortie souhaitées qui représentent les solutions.

3.3. Couches de neurones

18 Imaginez que vous êtes un peintre débutant et que vous désirez apprendre quelques techniques de base pour réaliser des paysages à l'huile. Un ouvrage pourrait vous enseigner une méthode progressive et simplifiée à l'extrême avec par exemple, ces quatre étapes : le dessin (on esquisse une ébauche grossière), la répartition des couleurs, l'affinement des traits, et la finition (lorsque les dernières touches sont apportées). L'important ici n'est pas le nombre d'étapes ou les particularités de chacune d'entre elles, mais le fait que l'ensemble du processus se déroule de manière incrémentielle, de telle sorte que la sortie d'une étape devienne l'entrée de la suivante ; chaque étape affinant le résultat précédent. Le résultat de la deuxième étape (répartition des couleurs) ressemble davantage à une véritable peinture de paysage que le résultat de la première (dessin). De même, le résultat de la quatrième étape (finition) peut être considéré, d'un point de vue conceptuel, comme meilleur que celui de n'importe laquelle des étapes précédentes.

19 En gardant cette analogie à l'esprit, il s'avère que le calcul neuronal bénéficie d'un processus incrémentiel similaire se déroulant étape par étape. Dans les années 60, des chercheurs ont en effet découvert qu'en intégrant plusieurs couches de neurones, des tâches plus complexes pouvaient être accomplies. Chaque couche d'un réseau de neurones multicouches affine la sortie de la couche précédente et fait un pas, plus ou moins grand, vers le résultat final. L'architecture résultante serait similaire à celle du schéma 3, mais avec un certain nombre de couches cachées supplémentaires. Cette structure en couches est clairement reconnaissable dans le réseau simplifié du schéma 3 où le calcul, effectué par deux couches, se déroule en deux étapes.

20 Un modèle composé de neurones organisés en couches est appelé réseau de neurones à couches. Malgré des résultats théoriques démontrant qu'un réseau à deux couches dispose d'une puissance de

calcul suffisante pour effectuer pratiquement n'importe quelle tâche, dans le monde réel, la puissance de calcul des réseaux de neurones semble être liée au nombre de couches. Les modèles comportant plus de quelques couches sont souvent qualifiés de réseaux de neurones profonds et les algorithmes d'entraînement correspondants sont appelés algorithmes d'apprentissage profond.

21 À titre d'exemple de la complexité que peuvent atteindre ces modèles profonds, GPT-3 (Brown *et al.* 2020), l'un des plus grands réseaux de neurones lancé en 2020 dans le domaine de la génération du langage naturel, comporte 96 couches comptant chacune des dizaines de milliers de neurones. Son algorithme d'entraînement a donc dû apprendre près de 175 milliards de poids. Plusieurs superordinateurs ont été utilisés pour cette tâche, un processus susceptible de prendre de quelques semaines à plusieurs mois. En effet, on estime que l'apprentissage des poids d'un tel modèle à l'aide d'un seul ordinateur *gaming* puissant aurait pris plus de 350 ans [1].

3.4. La traduction automatique neuronale

22 Dès lors que nous parvenons à représenter une phrase source comme un ensemble de données d'entrées pour un réseau de neurones, et que nous pouvons interpréter les données de sortie du réseau comme une phrase cible, nous disposons d'un système de *traduction automatique neuronale* (TAN). La TAN s'occupe tout d'abord de traiter les mots de la phrase source. Chaque mot source intégré par la partie *encodeur* du réseau de neurones entraîne des modifications des activations des ensembles de neurones spécifiques au sein de ce même réseau. Ensuite, lorsque la totalité de la phrase source a été traitée, la partie *décodeur* du réseau commence son travail, entraînée à attribuer, pas à pas, un score de probabilité pour chaque mot cible potentiel dans la traduction, en fonction des mots cibles qui ont déjà été générés. Bien que ce processus soit semblable au fonctionnement des claviers prédictifs des smartphones modernes, les prédictions de mots en TAN reposent également sur la phrase source, dans la mesure où elles sont supposées en être une traduction, comme nous le verrons un peu plus loin.

23 Les systèmes TAN sont donc des réseaux de neurones profonds dont l'architecture sera décrite ultérieurement, dans la section 6. Ils sont dotés de milliers de neurones et de millions de poids (parfois bien davantage) qui doivent être entraînés à l'aide d'exemples tirés d'un corpus parallèle constitué de millions de phrases sources et leurs traductions. Les représentations mathématiques des mots d'une phrase écrite dans la langue source servent de données d'entrée au réseau de neurones et les mots de la phrase correspondante dans la langue cible sont utilisés pour générer les données de sortie souhaitées. Comme vous pouvez vous en douter, l'entraînement d'un vaste réseau en un temps raisonnable est loin d'être évident. En effet, il faut disposer d'un équipement très performant et spécialisé dans le traitement des nombres pour exercer le réseau à partir d'exemples qui lui sont soumis à maintes et maintes reprises. À chaque itération, de légères modifications sont apportées aux poids du réseau afin d'améliorer sa prédiction de mots cibles.

3.5. L'entraînement des réseaux de neurones

24 Entraîner un réseau de neurones consiste à définir le poids des connexions entre ses neurones, de telle façon que, compte tenu d'une base d'apprentissage d'exemples de phrases sources-cibles, il génère des propositions réelles aussi proches que possible de celles des exemples correspondants.

25 Cet entraînement commence par un ensemble de poids aléatoires ou extraits d'un réseau de neurones résolvant une tâche similaire. Pendant cette phase d'apprentissage, les poids sont modifiés pour que la valeur de la fonction d'erreur ou fonction de perte (qui détermine le degré d'écart entre les données de sorties obtenues et celles souhaitées) soit aussi faible que possible. Les algorithmes d'entraînement (également appelés algorithmes d'apprentissage) calculent sans cesse de petites corrections (mises à jour) apportées aux poids jusqu'à ce que la fonction d'erreur soit minimale ou suffisamment faible pour tous les exemples de la base d'apprentissage, ou bien que des résultats satisfaisants soient obtenus avec une autre base de développement (voir section 7.2). Toutefois, les spécificités techniques de l'algorithme d'apprentissage dépassent le cadre de ce chapitre. Disons

simplement qu'en règle générale, celui-ci est basé dans un premier temps sur le calcul de la variation de la fonction d'erreur lorsque chaque poids est modifié par une valeur fixe et extrêmement faible (le gradient de la fonction d'erreur), puis dans un deuxième temps sur la légère variation de chaque poids dans la direction qui diminue cette fonction[2]. Cette méthode d'apprentissage, appelée *descente de gradient*, ne garantit aucunement que les meilleurs poids seront trouvés, mais de bons candidats seront probablement identifiés. L'intensité de ces variations de poids est régulée par un paramètre appelé *taux d'apprentissage*. Sa valeur est généralement plus élevée durant les premières étapes de l'entraînement de l'algorithme, mais son amplitude diminue progressivement à mesure que les poids se rapprochent de leur valeur finale. À noter que l'entraînement d'un réseau de neurones est assez laborieux : de nombreux exemples sont nécessaires et doivent lui être présentés à de nombreuses reprises pour être assimilés. Généralement, cette contrainte est plus liée aux limitations des algorithmes d'entraînement qu'au manque de capacité d'un réseau de neurones spécifique à trouver la solution à un problème.

26 Une fois la valeur des poids fixée, l'entraînement s'arrête (voir section 7.2) et le réseau de neurones peut être utilisé pour calculer les données de sorties de nouvelles entrées qui ne figurent pas parmi les exemples utilisés lors de son apprentissage.

3.6. La généralisation dans les réseaux de neurones

27 La *généralisation* est un processus cognitif fondamental chez l'être humain et l'animal. Elle nous permet d'utiliser ce que nous avons appris par le passé dans de nouvelles situations qui peuvent être perçues comme similaires, sans toutefois être identiques à la situation qui a donné lieu à l'apprentissage initial. Par exemple, une personne n'a pas besoin de réapprendre à conduire lorsqu'elle s'engage dans une nouvelle rue ou monte dans une nouvelle voiture. Ainsi, la généralisation survient lorsqu'un organisme répondant déjà à un stimulus donné d'une façon précise réagit à des stimuli similaires de manière semblable. C'est également un processus essentiel à

l'apprentissage des langues, à l'image des bébés qui apprennent rapidement à dire des phrases qu'ils n'ont jamais entendues auparavant.

28 Dans le contexte de la traduction automatique, les réseaux de neurones peuvent théoriquement généraliser en produisant des résultats analogues lorsqu'ils sont alimentés par des données d'entrées similaires, qu'elles aient été intégrées ou non dans la base d'apprentissage. En effet, l'une des particularités de ces systèmes TAN réside dans la *souplesse* de leurs calculs. Autrement dit, si les valeurs des données d'entrée sont légèrement modifiées, le résultat des formules de calcul ne changera pas de manière significative.

29 Globalement, pour aboutir à la généralisation, des phrases similaires doivent obtenir des représentations semblables. Or, comme ces représentations de phrases seront issues de représentations de mots, nous pouvons en conclure que la représentation de mots analogues par des nombres semblables est une condition préalable à la généralisation dans l'analyse neuronale du langage.

30 Dans la section suivante, nous verrons comment obtenir une liste appropriée de représentations neuronales pour les mots d'une phrase en s'appuyant sur la souplesse des réseaux de neurones, de sorte qu'après l'entraînement, le système est capable de généraliser correctement de nouvelles phrases jamais rencontrées auparavant.

4. Le plongement lexical : une représentation vectorielle des mots

31 Dans la section précédente, nous avons vu que les neurones sont généralement disposés en couches, de telle sorte que les données de sortie des neurones d'une couche deviennent les données d'entrée des neurones de la couche suivante. Il est intéressant de noter que les données de sortie du groupe de neurones d'une couche spécifique constituent une représentation de l'information qu'ils traitent à ce stade.

32 Dans le domaine du traitement automatique du langage naturel, les informations traitées par les réseaux de neurones, constituées de

mots et de leurs représentations au sein du réseau, sont généralement appelées *plongements* (Mikolov *et al.* 2013). La véritable utilité de ces *plongements* repose sur le fait que les mots ayant une signification similaire ou apparaissant habituellement dans les mêmes contextes finissent par avoir des plongements similaires. Afin de mieux comprendre ce processus, prenez une feuille de papier et dessinez un carré d'environ 10 centimètres de côté. À présent, prenez tous les mots de la liste ci-dessous et inscrivez-les sur la surface du carré en rapprochant les mots dont le sens est similaire. Si ce critère de proximité sémantique ne vous paraît pas assez précis, vous pouvez positionner les mots en fonction de leur fréquence de cooccurrence dans des phrases ou des paragraphes. Les mots sont les suivants : *restaurant*, *rouge*, *jardin*, *fontaine*, *fleur*, *tomate*, *ballon*, *serveurs*, *couteau*, *fleurs*, *menu*, *cuit*, *chromosome* et *constamment*. Faites-le avant de poursuivre votre lecture.

33 La restriction imposée à travers le critère de proximité sémantique des mots implique que vous n'avez pas pu les répartir librement sur le carré. Vous avez probablement décidé de regrouper des mots tels que *restaurant*, *menu* et *serveurs*, d'une part, et des mots tels que *jardin*, *fleurs* et *fontaine*, d'autre part. Il y a cependant des cas qui s'avèrent moins évidents : *rouge* est clairement un voisin de *tomate*, mais il devrait également être proche de *fleur*. Un compromis consisterait à le placer quelque part entre les deux, un peu plus près de *tomate* que de *fleur* si nous admettons que *rouge* n'est pas aussi essentiel aux fleurs qu'aux tomates.

34 Vous avez peut-être remarqué certains regroupements dans votre schéma : un îlot représentant le champ sémantique des restaurants et éléments connexes, et un autre îlot autour de l'idée de jardin et de verger. Il y a aussi quelques anomalies dans la liste. En particulier le mot *constamment* qui semble à priori déconnecté du reste des mots, nous obligeant à le placer aussi loin que possible des autres mots. *Chromosome* est un autre de ces mots isolés. Cependant, comme les fleurs et les serveurs utilisent des chromosomes pour transporter leurs informations génétiques, on peut donc le placer quelque part au milieu de la ligne entre ces deux mots, sans toutefois le mettre trop près de *rouge*. Reportez-vous au schéma 4 pour une proposition de solution qui ne correspondra pas nécessairement à la vôtre.

35 Afin d'attribuer des codes mathématiques aux mots de notre liste, assignons-leur maintenant des coordonnées qui reflètent leur position sur le carré. Comme nous sommes dans un espace bidimensionnel, chaque coordonnée sera constituée de deux nombres ; le premier représentant la distance au côté vertical gauche du carré, et le second, la distance au côté horizontal inférieur du carré. Par exemple, le mot *restaurant* pourrait se voir attribuer les deux nombres 0,25 et 1,1 ; et le mot *menu* (à proximité de *restaurant* cf. schéma 4.) les nombres 0,6 et 1,3. Les valeurs de ces coordonnées peuvent être exprimées en utilisant la *notation vectorielle* qui consiste simplement à écrire les nombres sous forme de liste de valeurs séparées par des points-virgules entre crochets. Les vecteurs correspondant à *restaurant* et *menu* seraient donc respectivement [0,25 ; 1,1] et [0,6 ; 1,3]. Chacun de ces vecteurs représente un plongement lexical possible pour ces deux mots.

36 Bien que cela ne soit pas forcément évident au premier abord, le fait que les plongements comportent deux nombres au lieu d'un seul augmente les possibilités de résoudre le problème du rapprochement ou de l'éloignement des mots, en nous donnant plus de liberté pour respecter toutes les restrictions. D'ailleurs, passer de deux à un nombre plus élevé de dimensions élargit encore le champ de ces possibilités. Ainsi, une représentation à cinq dimensions d'un mot pourrait ressembler à [2,34 ; 1,67 ; 4,81 ; 3,01 ; 5,61]. Les systèmes TAN traitent des plongements de plusieurs centaines de dimensions, et la phrase source à traduire est représentée par un ensemble de ces vastes plongements lexicaux.

37 Les plongements lexicaux sont appris à l'aide du même algorithme que celui utilisé pour l'apprentissage des poids du réseau de neurones présenté dans la section 3.5. Dans la pratique, il s'avère que les poids et les plongements sont appris simultanément. Puisque les plongements lexicaux ne sont rien d'autre que les activations d'un ensemble de neurones (ceux de la première couche du réseau de neurones), leur apprentissage revient aussi à connaitre les poids qui déterminent les activations de ces mêmes neurones.

4.1. La généralisation

38 Comme nous l'avons déjà évoqué, pour que le réseau soit capable de *généraliser* correctement, à savoir, apprendre à traduire et être capable de traduire des phrases jamais rencontrées au préalable, des phrases similaires doivent être représentées de manière analogue. Comme ces représentations de phrases sont obtenues à partir de plongements lexicaux, nous pouvons en conclure que la représentation de mots similaires à l'aide de nombres similaires est une condition préalable à la généralisation dans le traitement neuronal du langage naturel. En suivant notre exemple, des mots tels que *versé*, *plu*, *verser* ou *pleuvoir* devraient idéalement partager des plongements semblables puisqu'ils sont tous proches sur le plan sémantique. De même, les représentations mathématiques de *verser* et *pleuvoir* devraient être plus proches de mots tels que *conduire*, étant tous trois à l'infinitif et pouvant apparaître dans le même type de contexte ; tout comme *versé* et *plu* devraient être voisins étant tous deux des participes passés. C'est pour cette raison que nous avons en règle générale besoin de plusieurs dimensions : nous voulons que les mots soient proches les uns des autres de différentes façons ou pour différentes raisons, et ce simultanément.

4.2. Les propriétés géométriques des plongements lexicaux

39 Les plongements lexicaux présentent des propriétés intéressantes démontrant qu'ils représentent les caractéristiques sémantiques des mots (ou des propriétés liées à la sémantique). Comme nous l'avons expliqué, un plongement lexical est composé de plusieurs nombres réels, généralement des centaines voire des milliers, et chacun de ces nombres semble exprimer une partie du sens d'un mot. Par exemple, le plongement lexical pour *Dublin* devrait exprimer plusieurs de ses composantes sémantiques : une ville, la capitale d'Irlande, le lieu où se trouvent les sièges sociaux de plusieurs multinationales en Europe.

40 Grâce à cette spécialisation des différentes dimensions des plongements, il est possible d'effectuer quelques opérations arithmétiques et de parvenir à des résultats sensés. Ces opérations sont de simples additions et soustractions, aisément calculables. Additionner (ou

soustraire) deux plongements consiste simplement à additionner (ou soustraire) les nombres qui composent les vecteurs, un par un, par exemple [1,24 ; 2,56 ; 5,23] + [0,12 ; 1,12 ; 0,01] = [1,36 ; 3,68 ; 5,24]. Voici quelques exemples d'opérations arithmétiques donnant des résultats sensés et effectuées avec des plongements que les systèmes TAN apprennent habituellement :

> $[roi] - [homme] + [femme] \simeq [reine]$
> $[Dublin] - [Irlande] + [France] \simeq [Paris]$.

41 où les crochets désignent des plongements lexicaux et ≃ exprime que le plongement résultant de l'opération est proche du plongement lexical dans la partie droite de l'exemple. Cela peut s'interpréter comme une indication que *roi* soit pour *homme* ce que *reine* est pour *femme*, un monarque ou une souveraine ; et *Dublin* est pour *Irlande* ce que *Paris* est pour *France*, la capitale d'un pays.

5. Les plongements lexicaux contextuels et l'attention

42 Le sens des mots varie d'une phrase à l'autre. Le plongement lexical de *lettre*, par exemple, devrait être différent selon si le mot fait référence au caractère alphabétique ou à un document adressé à une autre personne.

Schéma 4



Positionnement de mots dans une aire bidimensionnelle de manière à ce que les mots apparentés soient proches les uns des autres, mais loin des mots avec lesquels ils ont moins en commun.

43 Il pourrait d'ailleurs être intéressant pour un système TAN de représenter le mot par des plongements lexicaux différents selon s'il s'agit d'une lettre d'amour ou d'une lettre de plainte. Les plongements présentés jusqu'ici sont *non-contextuels* : ils ont été effectués en prenant en considération des cooccurrences de mots qui appa-

raissent fréquemment dans les phrases, mais sans tenir compte des différents sens que les mots peuvent porter.

44 Dans l'arène de la TAN, *l'attention* joue un rôle important, car elle permet au réseau de neurones d'effectuer des *plongements lexicaux contextuels*, c'est-à-dire, des représentations vectorielles des mots d'une phrase, calculées de façon à ce que la représentation obtenue pour un mot soit adaptée à son sens au sein de chaque phrase. L'attention étant, souvenez-vous, un concept mis en place au moyen d'opérations mathématiques commodément apprises par un algorithme d'entraînement. Dans notre contexte, l'attention peut être comparée à la situation dans laquelle nous accordons notre attention à quelque chose ou quelqu'un dans notre vie quotidienne.

45 En mettant à profit l'attention pour se concentrer sur certains mots dans la phrase, le vecteur du plongement correspondant par exemple au mot *saison*, présentera des différences entre la phrase « Le premier épisode va continuer directement là où la saison précédente s'est arrêtée » et la phrase « L'été est la saison la plus chaude de l'année ». En théorie, il peut sembler que le but des plongements lexicaux contextuels est d'obtenir des représentations qui correspondent aux différents sens d'un mot, mais, bien que cela soit en général le cas, l'idée ne s'y limite pas. Les plongements lexicaux pour *saison* dans les phrases « L'hiver est la saison la plus froide de l'année dans les zones polaires et tempérées », « L'été est la saison la plus chaude de toute l'année » et même « De toute l'année, c'est l'été qui est la saison la plus chaude » seront tous différents, bien que certainement plus proches les uns des autres que de la représentation de *saison* dans « Le premier épisode reprendra exactement là où la saison précédente s'est arrêtée ». Ces divergences proviennent du fait que l'ordre des mots dans les phrases ou bien les mots eux-mêmes sont différents. Il est à noter que, dans chacun de nos exemples, les deux instances de *la* obtiendront deux vecteurs contextuels différents, car le contexte est également différent pour chaque instance.

46 Comment les plongements lexicaux contextuels sont-ils calculés grâce à l'attention ? En prenant la phrase « La saison la plus chaude est toujours l'été », la procédure commence par la récupération des plongements lexicaux non-contextuels introduits dans la Section 4. La phrase étant composée de neuf mots, le résultat est un ensemble

de neuf vecteurs qui sont les ingrédients de la prochaine étape. Maintenant, afin de calculer le plongement lexical contextuel pour le mot *saison* dans la phrase, un vecteur d'attention est généré mathématiquement par le réseau de neurones. Ce vecteur d'attention est composé de neuf pourcentages représentant le degré d'attention devant être accordé à chacun des mots dans la phrase, ce dans le but d'obtenir la représentation du mot *saison*. L'élément à une position donnée dans le vecteur correspond à l'attention accordée au mot à cette même position dans la phrase. Par exemple, un vecteur d'attention [10 % ; 25 % ; 0 % ; 7 % ; 15 % ; 8 % ; 8 % ; 2 % ; 25 %] indiquerait que, afin de calculer un plongement lexical contextuel du mot *saison* dans cette phrase, les plongements lexicaux pour *été* et *saison* seront tout aussi importants (ils reçoivent ensemble 50 pour cent de l'attention totale), ce qui est logique, car ils sont sémantiquement liés au concept de la saison météorologique. Veuillez remarquer que le déterminant qui précède reçoit également un peu d'attention (10 %), ce qui peut être expliqué par le fait qu'il aide à catégoriser *saison* comme un nom. La contribution du verbe (8 %) au plongement lexical contextuel peut également être décrite selon sa contribution à marquer *saison* comme étant au singulier. Notez que la somme des pourcentages donne toujours 100 %.

47 Déterminer comment le vecteur d'attention est utilisé afin d'obtenir un nouveau plongement combinant les plongements originaux pour créer un nouveau plongement dépasse le cadre de ce chapitre. Sachez simplement que la procédure implique une séquence spécifique d'opérations mathématiques et que le plongement ainsi obtenu se trouve quelque part entre les plongements originaux.

48 Pour reprendre notre exemple, neuf vecteurs d'attention sont calculés pour cette phrase (un pour chaque mot) puis appliqués aux plongements lexicaux non-contextuels originaux afin d'obtenir un ensemble de neuf nouveaux plongements, chacun correspondant à un mot différent de la phrase. Ces nouveaux plongements sont considérés comme étant des plongements lexicaux contextuels, étant influencés à divers degrés par le reste des mots de la phrase.

5.1. Plusieurs couches d'attention valent mieux qu'une

49 Nous avons abordé précédemment, dans la section 3.3 de ce chapitre, les avantages de l'amélioration successive des calculs neuronaux à l'aide de modèles composés de différentes couches de neurones. Par conséquent, il n'y a rien de très étonnant à ce que les plongements lexicaux contextuels tout juste obtenus puissent être combinés à de nouveaux vecteurs d'attention pour obtenir, une fois de plus, un autre nouveau plongement pour chaque mot, et arriver à des représentations plus précises. Si l'on prend un exemple concret, Turing-NLG, un des plus grands modèles de langue publié en 2020, possède 78 couches d'attention affinant progressivement des plongements de 4256 dimensions [3]. Rappelez-vous que ces représentations, apprises par l'application de multiples couches consécutives, sont connues sous le nom de représentations *profondes*

5.2. Plusieurs têtes valent mieux qu'une

50 Il n'y a aucune raison de se restreindre à un seul vecteur d'attention pour chaque mot dans chaque couche. Par exemple, si l'on considère la phrase « Mon papy cuit du pain au four quotidiennement », il pourrait être intéressant d'avoir un plongement lexical pour *four* possédant la nuance de *papy* afin de refléter que le four appartient à une personne âgée, et un plongement différent pour *four* avec la nuance de *pain* pour refléter ce qui a été cuit dedans. Un unique vecteur d'attention serait obligé de mélanger ces deux nuances dans un seul plongement lexical contenant des informations trop hétérogènes, ce qui pourrait avoir un impact négatif sur la recherche d'une traduction pour le mot représenté par le plongement. Pour cette raison, certains systèmes de TAN génèrent différentes attentions pour chaque mot dans chaque couche de neurones, puis les utilisent pour calculer un certain nombre de plongements différents pour chaque mot. On dira que chacun de ces plongements est calculé par une *tête* différente. Turin-NLG possède 28 têtes d'attention dans chaque couche. Ainsi, sa dernière couche génère 28 plongements lexicaux différents comportant 4256 dimensions pour chaque mot.

5.3. Les plongements lexicaux contextuels et le traitement automatique du langage naturel

51 Les plongements sont la clé de voûte de la TAN, mais ils se sont également révélés utiles dans bien d'autres applications du traitement du langage naturel. Pour illustrer cela, les systèmes qui classent automatiquement comme positives ou négatives les phrases d'un texte portant sur un test de produit peuvent fonctionner en calculant dans un premier temps un ensemble de plongements lexicaux contextuels profonds pour chaque mot de la phrase. Ces plongements sont ensuite utilisés pour alimenter un réseau de neurones bien plus simple qui va alors calculer un nombre entre 0 et 1 indiquant le degré de positivité de la phrase (ainsi, 0,95 indiquera une phrase résolument positive, 0,2 une phrase négative et 0,51 une phrase neutre). Ces systèmes sont généralement entraînés avec des corpus de phrases balisés manuellement par des humains. La partie de ce modèle chargée de calculer les plongements lexicaux n'est pas forcément entraînée pour un corpus en particulier, car des modèles *pré-entrainés* avec des millions de phrases sont disponibles gratuitement dans de nombreuses langues.

6. Et enfin, la traduction automatique neuronale

52 Vous êtes désormais, espérons-le, en bonne posture pour comprendre le fonctionnement de la TAN, même si nous devons rester succincts dans la description de ses aspects fondamentaux. Nous allons maintenant nous concentrer sur deux architectures de réseaux de neurones : le modèle Transformer et le modèle récurrent.

6.1. Le modèle Transformer : un encodeur-décodeur basé sur l'attention

53 Pour faire simple, un système TAN Transformer est composé d'un module calculant des plongements lexicaux contextuels pour chacun des mots de la phrase source et d'un deuxième module prédisant

successivement chaque mot de la phrase cible. Le premier module est appelé l'*encodeur* et le deuxième module est appelé le *décodeur*. Pour prédire les mots dans la langue cible, le décodeur fait attention aux plongements de tous les mots de la phrase source ainsi qu'aux plongements des mots cibles déjà générés. L'architecture complète est appelée un *Transformer* (Vaswani *et al.* 2017). Le schéma 5 nous montre l'exemple d'un encodeur à trois couches et des degrés d'attention pris en compte afin de calculer un plongement dans la deuxième et la troisième couche. Le schéma 6 représente un encodeur dans un diagramme étendu afin d'y inclure également le décodeur et de montrer l'architecture complète du modèle Transformer.

54 Un corpus parallèle est utilisé par l'algorithme d'apprentissage afin d'obtenir une série de poids, de plongements et de vecteurs d'attention pour le Transformer, de sorte que les données d'entraînement puissent être reproduites jusqu'à un certain degré et que le système puisse généraliser au-delà des phrases de la base d'apprentissage.

55 Par exemple, considérons qu'un Transformer avec une seule tête par couche est utilisé pour traduire la phrase « Mon papy cuit du pain au four quotidiennement » en espagnol.

Schéma 5



L'encodeur d'un système de traduction automatique neuronal de type Transformer. Le symbole *start* est généralement préfixé afin de marquer explicitement le début de la phrase. Le diagramme montre également que les plongements dans la première couche de *grand* et *renard* contribuent à divers degrés à l'obtention des plongements de *renard* dans la deuxième couche ; de même, le plongement pour *grand* dans la dernière couche intègre des informations provenant de tous les plongements de la deuxième couche en utilisant différents degrés d'attention.

56 L'encodeur génère d'abord un ensemble de huit vecteurs de plongement. Le décodeur calcule ensuite un vecteur d'attention à huit dimensions tel que [60 % ; 10 % ; 0 % ; 0 % ; 0 % ; 30 % ; 0 % ; 0 %] et l'utilise afin d'obtenir une nuance de la phrase source lui permettant de générer un plongement pour le premier mot de la phrase cible. Partons du postulat que le système génère correctement le mot espagnol *mi*. Le décodeur calcule ensuite un vecteur d'attention à neuf dimensions tel que [50 % ; 10 % ; 0 % ; 0 % ; 0 % ; 20 % ; 0 % ;

0 % ; 20 %] (le dernier pourcentage correspond à l'attention accordée au premier mot de la phrase cible) et l'utilise afin de produire un plongement du deuxième mot dans la phrase cible. Cette procédure continue jusqu'à ce que le décodeur génère une marque spéciale indiquant la fin de la phrase.

57 Le vecteur produit par le décodeur à chaque étape n'est pas tout à fait une estimation du plongement du mot suivant. En réalité, une couche supplémentaire est ajoutée à la fin du décodeur pour calculer un vecteur de probabilité pour chaque mot dans le vocabulaire de la langue cible. La section 7.3 va vous montrer comment ces probabilités peuvent être utilisées afin d'obtenir la séquence de mots qui constitue la phrase dans la langue cible.

Schéma 6



Un système de traduction automatique neuronale complet de type Transformer traduisant une phrase. Une version élargie de l'encodeur est affichée dans le schéma 5. Notez que la prévision de *zorro* est non seulement obtenue en accordant de l'attention aux plongements des mots cibles précédents, mais également aux plongements correspondants à certains des mots en provenance de la dernière couche de l'encodeur.

6.2. Architecture récurrente

58 Même si le modèle Transformer présenté dans la section précédente est actuellement utilisé dans la plupart des systèmes TAN commerciaux, d'autres modèles existent tels que le très populaire encodeur-décodeur *récurrent* (Bahdanau *et al.* 2015). De façon similaire au modèle Transformer, un encodeur génère un ensemble de plongements pour les mots de la phrase source. Un décodeur utilise ensuite l'attention pour calculer les plongements de chaque mot de la phrase cible en intégrant les informations des mots de la phrase source et des mots déjà générés pour la phrase cible. Néanmoins, l'encodeur et le décodeur du modèle récurrent calculent les plongements lexicaux contextuels de manière locale. Par exemple, les plongements du cinquième mot encodé sont basés d'une part, sur les plongements des quatre premiers mots, et d'autre part, sur les plongements des mots suivants. Ce calcul est réalisé en parcourant la phrase source de

gauche à droite et de droite à gauche. Le schéma 7 est un diagramme de ce modèle montrant uniquement le traitement de gauche à droite.

59 Il est intéressant de noter que le modèle mathématique utilisé restreint l'importance accordée aux mots entourant la cible du plongement lexical contextuel (dans notre exemple le cinquième mot).

Schéma 7



Sous-modèle *gauche à droite* de l'encodeur d'un système de traduction automatique neuronale récurrent, juste après le traitement de « <start> Le grand » et sur le point de traiter « renard ».

60 En résulte un mécanisme qui a tendance à ignorer les représentations des mots distants et se concentre particulièrement sur les mots les plus proches. Similairement au modèle Transformer, une couche finale à la fin du décodeur va, pour chaque mot de la langue cible, calculer un vecteur donnant la probabilité que ce mot soit dans une position correspondante à celle de son équivalent dans la phrase source. Forcada (2017) décrit de manière plus détaillée le modèle encodeur-décodeur récurrent et présente également les types de résultats que la TAN produit.

7. Paramètres supplémentaires

7.1. Mots et sous-mots

61 Comme nous l'avons expliqué dans ce chapitre, un plongement est calculé pour chaque mot suite à la phase d'entraînement, indépendamment de l'utilisation d'un modèle récurrent ou Transformer. Cela veut-il dire que l'on finit par obtenir un plongement pour chaque mot possible de la langue ? Pas tout à fait. Les langues, particulièrement celles fortement fléchies ou agglutinées, peuvent aisément posséder des centaines de milliers voire des millions de formes lexicales différentes. Afin de comprendre pourquoi cela représente un défi pour les systèmes TAN, gardez à l'esprit que la somme des plongements lexicaux (désigné comme le *vocabulaire*) exerce une influence sur le nombre de poids dans le réseau de neurones et que les grands

réseaux de neurones rencontrent souvent des difficultés à généraliser vers des données inconnues. La taille du vocabulaire pourrait être réduite en prenant uniquement en compte les formes lexicales présentes dans la base d'apprentissage, mais cela implique généralement de prendre en considération un nombre de mots conséquents et soulève un nouveau problème. Lorsque la phase d'entraînement est terminée et que le système de TAN traduit de nouvelles phrases contenant des mots absents de la base d'apprentissage, ces mots inconnus déstabilisent le fonctionnement du modèle et lui font perdre en précision. Un plongement non-contextuel spécifique, réservé à cette situation, étant alors attribué à chaque mot inconnu.

62 Les ingénieurs sont donc arrivés à la solution suivante : diviser les mots en *sous-mots*. Idéalement, ces sous-mots devraient avoir une logique linguistique et comporter certaines composantes sémantiques, par exemple, diviser *démystifiant* en *dé– + – myst– + – ifi– + – ant* est sûrement plus logique linguistiquement parlant (et probablement plus utile pour la traduction automatique) que le diviser en *dém– + – ystif– + –ia– + –nt*. Cependant, effectuer une division sensée du point de vue de la linguistique requiert l'existence d'un ensemble de règles et procédures de division pour la langue concernée, une ressource qui n'est pas forcément disponible pour toutes les langues.

63 Ce problème est généralement contourné par l'apprentissage automatique des règles de division au moyen de l'examen attentif de vastes textes, contenant par exemple toutes les phrases sources ou cibles de la base d'apprentissage. Une approche répandue [4] appelée *codage par paire d'octets* (Sennrich *et al.* 2016), commence avec la combinaison des lettres en sous-mots de deux lettres, trois lettres, etc. lorsqu'ils apparaissent fréquemment dans le corpus [5]. Le codage par paire d'octets identifierait et isolerait certainement un suffixe *–ant*, fréquent dans de nombreuses formes verbales (*marchant*, *considérant*), même dans le cas de formes inconnues (comme *asterixlegauloisant*) ; *–ant* serait ensuite transformé en un plongement contextuel contenant sa composante sémantique.

7.2. Critères d'arrêt et d'évaluation

64 En plus d'une grande base d'apprentissage, comme mentionné précédemment dans la section 3.5, un *corpus de développement* plus

modeste est généralement employé à part, à d'autres fins que l'entraînement. Le but de ce corpus est de surveiller les performances du système de TAN durant son entraînement, puis de décider, par exemple, quand celui-ci doit s'arrêter. La phase d'entraînement essaie de minimiser la fonction d'erreur (ou dans le cadre de la TAN, de maximiser la probabilité des phrases cibles de la base d'apprentissage). L'un des problèmes pouvant survenir vient du fait qu'un entraînement trop intense à l'aide de la base d'apprentissage peut entraver la généralisation, car le réseau de neurones peut en arriver à trop *mémoriser* les exemples de traduction. C'est alors que le corpus de développement entre en jeu : après un certain nombre d'itérations, ou étapes de l'algorithme d'entraînement, ses phrases sources sont traduites par le réseau de neurones. Le résultat est automatiquement comparé aux phrases cibles souhaitées de ce même corpus en utilisant des critères d'évaluation automatiques approximatifs simples, dont le plus commun est BLEU (Papineni *et al.* 2002). BLEU mesure combien de suites d'un, deux, trois et quatre mots présentes dans le résultat sont retrouvées dans la phrase cible de référence, et calcule alors un score allant de 0 (pas d'équivalence) à 100 % (toutes les suites sont retrouvées). Lors de la phase d'entraînement, le score BLEU de la base de développement peut indiquer une dégradation des performances. Dans ce cas, l'entraînement peut être arrêté, ou bien l'ensemble de poids en cours d'utilisation peut être stocké avant de poursuivre l'entraînement pendant un certain temps pour voir si BLEU indique une amélioration. Il existe bien entendu beaucoup d'autres critères d'évaluation automatique pouvant jouer le rôle de BLEU dans ce processus.

7.3. Recherche en faisceau

65 Dans les systèmes de TAN, le décodeur génère la phrase cible de façon séquentielle, un mot à la fois, comme nous l'avons expliqué dans les sections 6.1 et 6.2. À chaque étape, le réseau de neurones calcule une probabilité (comprise entre 0 et 100 %) pour chaque mot du vocabulaire cible. Une façon d'utiliser cette information est de choisir le mot le plus probable et de l'employer dans la phrase cible en ignorant les autres possibilités. Remarquez que, procéder de la sorte détermine la marche à suivre pour le système de TAN, car la prédiction en cours est utilisée comme donnée d'entrée par le déco-

deur dans l'étape suivante (par exemple, le mot *zorro* dans le schéma 6). Une manière d'explorer davantage de possibilités est de sélectionner, par exemple, les trois mots les plus probables, et de *cloner* le système en trois, chaque système étant déterminé respectivement par l'un des trois mots, puis de voir les résultats proposés. Mais cette méthode ne saurait être utilisée indéfiniment, car le nombre de systèmes traduisant la phrase triplerait à chaque étape et augmenterait exponentiellement. Afin d'éviter cette situation, seul un certain nombre de systèmes ont le droit de *survivre* : ceux qui obtiennent la valeur la plus élevée dans un calcul approximatif de la probabilité de la phrase complète restant à traduire. Cette méthode appelée *recherche en faisceau* est une approximation communément retrouvée dans d'autres modèles probabilistes de traitement automatique du langage humain telle que la reconnaissance vocale.

Conclusions

66 Afin d'entraîner un système de TAN, des milliers voire des millions d'exemples de paires de phrases sources-phrases cibles sont nécessaires. Pour beaucoup de paires de langues, de domaines et de genres textuels, de telles ressources n'existent pas. La TAN est donc inutilisable pour beaucoup d'usages spécifiques. En revanche, pour des langues disposant d'importantes ressources, la TAN à usage général est très communément utilisée, et pas seulement par les traducteurs.

67 Ce chapitre a introduit avec un certain degré de technicité les éléments clés des systèmes TAN, et a exploré comment ils interagissent au sein des deux architectures les plus populaires actuellement, à savoir les modèles de réseaux de neurones Transformers et récurrents. À l'heure où ce livre est écrit, la recherche dans ce domaine est si soutenue que de nouveaux prototypes de modèles apparaissent quasiment chaque mois. Les modèles Transformers sont actuellement le paradigme préféré si un nombre suffisant de corpus parallèles est disponible pour les entraîner. En effet ils permettent des améliorations de qualité subtiles en comparaison avec les réseaux de neurones récurrents et le temps nécessaire à leur entraînement est plus court, mais un retournement de situation peut se produire à tout instant.

Bahdanau, Dzmitry, Kyunghyun Cho et Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. Dans Yoshua Bengio et Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015*.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Pra – fulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Dario Amodei, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever et Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:*2005.14165.

Forcada, Mikel L. 2017. Making sense of neural machine translation. *Translation Spaces* 6(2). 291–309.

Goodfellow, Ian, Yoshua Bengio et Aaron Courville. 2016. *Deep learning*. MIT Press.

Kudo, Taku et John Richardson. 2018. SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. Bruxelles, Belgique : Association for Computational Linguistics.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado et Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, vol. 30, 3111–3119.

Papineni, Kishore, Salim Roukos, Todd Ward et Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318.

Sennrich, Rico, Barry Haddow et Alexandra Birch. 2016. Neuralmachine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (volume 1: long papers)*,1715–1725. Berlin, Allemagne : Association for Computational Linguistics

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser et Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30, 5998–6008. Curran Associates, Inc.

1 “OpenAI's GPT-3 language model: a technical overview” (2020). Extrait de https://lambdalabs.com/blog/demystifying-gpt-3 (en anglais).

2 Certains d’entre vous reconnaîtront peut-être ici le concept mathématique de *dérivée d’une fonction*.

3 “Turing-NLG: A 17-billion-parameter language model by Microsoft”, 2020. Récupéré sur https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

4 Il existe des méthodes plus avancées telles que *SentencePiece* (Kudo et Richardson 2018), qui traite le texte entier comme une séquence de caractères et effectue la division en mots (*tokenization*) et sous-mots d’une seule traite.

5 Le codage par paire d’octets était au départ un algorithme de compression de texte : les séquences de lettres fréquentes (*octets*) étaient stockées une fois et remplacées par de courtes lignes de code pour réduire l’espace de stockage nécessaire.

Français
Dans ce chapitre, nous présentons dans ses grandes lignes les grands principes de bases des systèmes de traduction automatique neuronale. Nous introduisons progressivement les concepts de *réseau de neurones*, d’*algorithme d’apprentissage*, de *plongement lexical*, d’*attention* et d’*architecture encodeur-décodeur* utilisés pour décrire ces systèmes, afin de permettre à nos lectrices et à nos lecteurs d’obtenir une vue d’ensemble de leur fonctionnement interne et des possibilités qu’ils ont à offrir.

Juan Antonio Pérez-Ortiz
Université d’Alicante, Espagne

Mikel L. Forcada
Université d’Alicante, Espagne

Felipe Sánchez-Martínez
Université d’Alicante, Espagne

Sébastien Palmieri
Étudiant du master TSM de l'université Grenoble-Alpes

Romain Revet
Étudiant du master TSM de l'université Grenoble-Alpes