

## Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique

Option : Systèmes Informatiques

---

## Création d'un corpus de l'aphasie de Broca et développement d'un système Speech-to-speech de réhabilitation de la parole

---

Réalisé par :

BELGOUMRI Mohammed  
Djameleddine  
[im\\_belgoumri@esi.dz](mailto:im_belgoumri@esi.dz)

*Encadré par :*

Pr. SMAILI Kamel  
[smaili@loria.fr](mailto:smaili@loria.fr)  
Dr. LANGLOIS David  
[david.langlois@loria.fr](mailto:david.langlois@loria.fr)  
Dr. ZAKARIA Chahnez  
[c\\_zakaria@esi.dz](mailto:c_zakaria@esi.dz)

# Table des matières

<b>Page de garde</b>	<b>i</b>
<b>Table des matières</b>	<b>ii</b>
<b>Table des figures</b>	<b>iv</b>
<b>Sigles et abréviations</b>	<b>v</b>
<b>1 Notions générales</b>	<b>1</b>
1.1 Aphasicie de Broca . . . . .	1
1.1.1 Histoire et anatomie de l'aphasicie . . . . .	1
1.1.2 Classification des syndromes aphasicie . . . . .	4
1.1.3 Causes, prévalence et incidence de l'aphasicie . . . . .	4
1.1.4 Effet de l'aphasicie de Broca sur la qualité de vie . . . . .	5
1.2 Solution proposée . . . . .	6
1.2.1 Automatisation de la rééducation de la parole . . . . .	6
1.2.2 Structure proposée pour la solution . . . . .	7
1.3 Conclusion . . . . .	8
<b>2 Apprentissage séquence-à-séquence</b>	<b>9</b>
2.1 Énoncé du problème . . . . .	9
2.2 Perceptorns multicouches . . . . .	10

2.2.1	Généralités . . . . .	10
2.2.2	Application à la modélisation de séquence . . . . .	11
2.2.3	Avantages et inconvénients . . . . .	12
2.3	Architecture encodeur–décodeur . . . . .	12
2.4	Réseaux de neurones récurrents . . . . .	13
2.4.1	Réseaux de neurones récurrents simples . . . . .	14
2.4.2	Portes, gated recurrent unit et long short-term memory . . .	15
2.5	Réseau de neurones à convolutions . . . . .	18
2.6	Transformateurs . . . . .	21
2.6.1	Architecture générale . . . . .	21
2.6.2	Encodage positionnel . . . . .	21
2.6.3	Couches attention . . . . .	22
2.6.4	Autres éléments de l'architecture . . . . .	24
2.6.5	Analyse comparative de la performance . . . . .	25
2.7	Conclusion . . . . .	25
<b>3</b>	<b>Traduction automatique et reconnaissance automatique de la parole</b>	<b>26</b>
3.1	Traduction automatique . . . . .	26
3.1.1	Traduction automatique à base de transformeur . . . . .	27
<b>Bibliographie</b>		<b>33</b>

# Table des figures

1.1	Cerveau de Victor Louis Leborgne avec la lésion encadrée . . . . .	2
1.2	Encéphale humain. . . . .	3
1.3	Division morphologique et fonctionnelle du cerveau. . . . .	3
1.4	Comparaison de la qualité de vie de communication chez les individus saints et ceux qui souffrent de l'aphasie de Broca. . . . .	5
1.5	Diagramme de fonctionnement du système proposé. . . . .	6
1.6	Étapes intermédiaires du système proposé. . . . .	7
2.1	Architecture sous-jacente d'un MLP de profondeur 4. . . . .	10
2.2	Architecture encodeur-décodeur . . . . .	13
2.3	RNN v.s FFN . . . . .	13
2.4	Dépliement temporel d'un RNN sur une entrée de longueur 4. . . .	14
2.5	Dépliement temporel d'un encodeur-décodeur récurrent. . . . .	15
2.6	Forme générale d'un RNN à portes. . . . .	16
2.7	Architecture interne d'un GRU . . . . .	16
2.8	Architecture interne d'un LSTM . . . . .	17
2.9	Couche convolutive unidimensionnelle. . . . .	19
2.10	Recurrent Continuous Translation Model . . . . .	20
2.11	Architecture de ByteNet. . . . .	20
2.12	Architecture de ConvS2S. . . . .	20

2.13 L'architecture de transformeur. . . . .	22
2.14 Matrice d'encodage de la position pour $r = 10^4$ et $d = 512$ . . . . .	23
2.15 Schéma d'une couche attention multitête. . . . .	24
3.1 L'architecture de transformeur. . . . .	28

# Sigles et abréviations

ASR	reconnaissance automatique de la parole
AVC	accident vasculaire cérébrale
BPE	byte pair encoding
BPTT	rétro-propagation dans le temps
CNN	réseau de neurones à convolutions
DL	apprentissage profond
FFN	réseau de neurones feed-forward
GRU	gated recurrent unit
LSTM	long short-term memory
ML	apprentissage automatique
MLP	perceptron multicouches
MT	traduction automatique
NLP	traitement automatique du langage
NMT	traduction automatique neuronale
RMBT	traduction automatique à base de règle
RNN	réseau de neurones récurrent
S2S	séquence-à-séquence

# Chapitre 1

## Notions générales

Dans ce chapitre, nous traçons les grandes lignes de notre étude. Nous commençons par introduire les détails du problème principal que nous abordons (à savoir l'aphasie de Broca). Ensuite, nous présentons l'architecture globale de la solution que nous proposons. Dans les deux chapitres suivent, nous explorons la littérature sur les méthodes possibles pour implémenter notre solution.

### 1.1 Aphasie de Broca

L'aphasie ; emprunté au Grec ancien  $\alpha\phiασία$  qui veut dire “mutisme”, est un trouble de communication d'origine neurologique (LAROUSSE, s. d.). Elle affecte la capacité à comprendre le langage, s'y exprimer ou les deux. L'aphasie n'est pas causée par un trouble moteur, sensoriel, psychique ou intellectuel (CHAPEY, 2008). Sa cause principale est un accident vasculaire cérébral (AVC), mais elle peut également être le résultat d'une infection ou tumeur cérébrale, un traumatisme crânien, un trouble métabolique comme le diabète ou une maladie neurodégénérative comme l'Alzheimer (HALLOWELL, 2017).

#### 1.1.1 Histoire et anatomie de l'aphasie

Louis Victor Leborgne, né en 1809 à Moret-sur-Loing commence à perdre la capacité de parler à l'âge de 30 ans. Il est admis à l'hôpital de Bicêtre où il passerait 21 ans pendant lesquelles, il ne communiquait qu'en produisant le son “tan”, typiquement répété deux fois, si bien qu'on lui a donné le surnom “monsieur Tan tan” (MOHAMMED et al., 2018).

Le 11 avril 1861, monsieur Leborgne est examiné par Dr. Pierre Paul Broca pour une gangrène dans son pied droit. Dr. Broca s'intéresse au trouble linguistique dont souffre son patient (LORCH, 2011). Il fait l'observation que les facultés intellectuelles et motrices de monsieur Leborgne sont intactes, il en conclut qu'elles ne peuvent être à l'origine de son handicap. Dr. Broca donne le nom “aphémie” à ce type de situation (BROCA, 1861), il en écrit :

“Cette abolition de la parole, chez des individus qui ne sont ni paralysés ni idiots, constitue un symptôme assez singulier pour qu'il me paraisse utile de la désigner sous un nom spécial. Je lui donnerai donc le nom d'aphémie ( $\alpha$  privatif; φημι, je parle, je prononce); car ce qui manque à ces malades, c'est seulement la faculté d'articuler les mots.”

— BROCA, 1861.

Dr. Broca prend ce constat comme confirmation de ce qu'il appelait “le principe de localisations cérébrales”. Il s'agit de l'idée que le cerveau fonctionne comme système à plusieurs composants plutôt qu'un monolithe et que les fonctions cognitives sont spatialement localisées (FODOR, 1983).

Quand monsieur Leborgne est décédé le 17 avril, Dr. Broca lui fait l'autopsie. En ouvrant le crâne, il observe une lésion dans le cortex inférieur gauche du lobe frontal (voir Figure 1.1). Il en déduit que (1) cette lésion était à l'origine de l'aphémie de monsieur Leborgne et que (2) la partie affectée du cerveau est responsable d'articuler des expressions dans le langage (BROCA, 1861 ; LORCH, 2011 ; MOHAMMED et al., 2018).

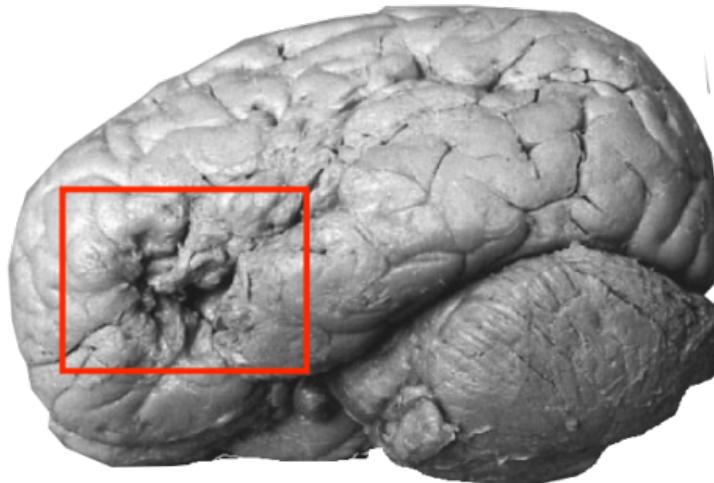


FIGURE 1.1 – Cerveau de Victor Louis Leborgne avec la lésion encadrée

Le trouble que Dr. Broca appelle “aphémie” est aujourd’hui connu sous le nom d’aphasie de Broca. Le cas de monsieur Leborgne est largement reconnu comme le premier cas enregistré d’aphasie en général et d’aphasie de Broca en particulier (MOHAMMED et al., 2018).

La quête de comprendre l’aphasie en général et celle de Broca en particulier, commence avec le cerveau. Celui des humains est le système le plus complexe connu (SCIENCES et al., 1992). Avec le cervelet et le tronc cérébral, il forme l’encéphale (voir Figure 1.2). Le cerveau se charge du traitement des flux nerveux sensoriels et moteurs. Il est aussi le siège des hautes fonctions cognitives comme l’inférence logique, l’émotion et — crucialement pour notre étude — le traitement du langage (FODOR, 1983).

Le cerveau est composé de deux hémisphères ; chacun desquels se divise en lobes : frontal, temporal, pariétal et occipital (voir Figure 1.3a). La surface du cerveau s’appelle

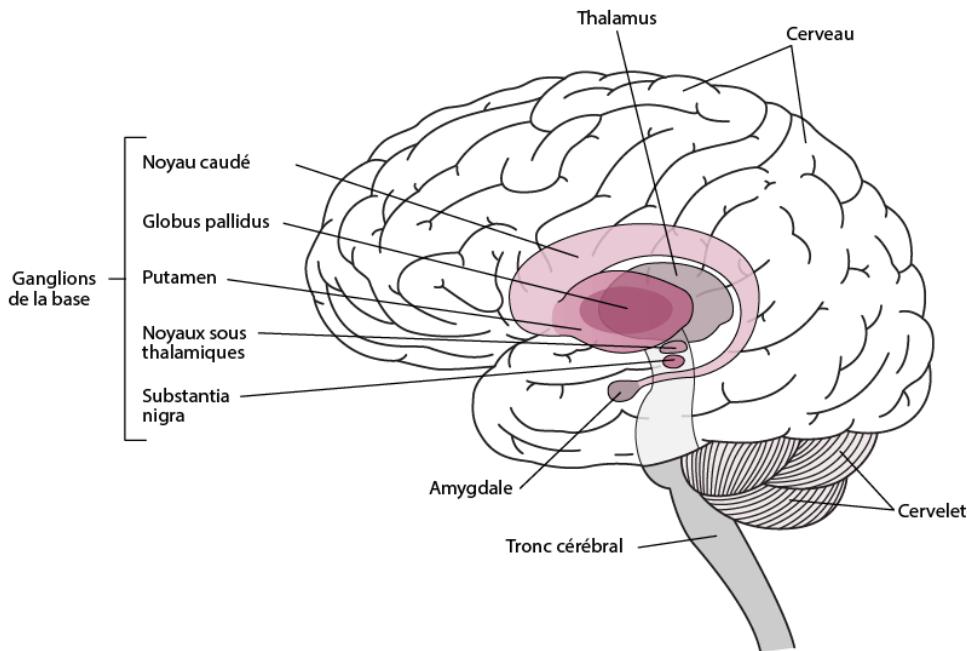
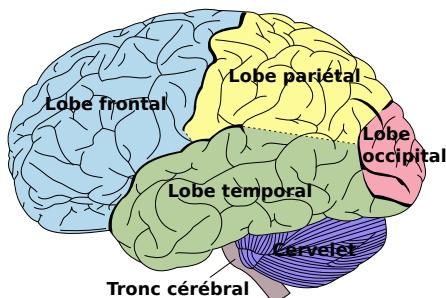
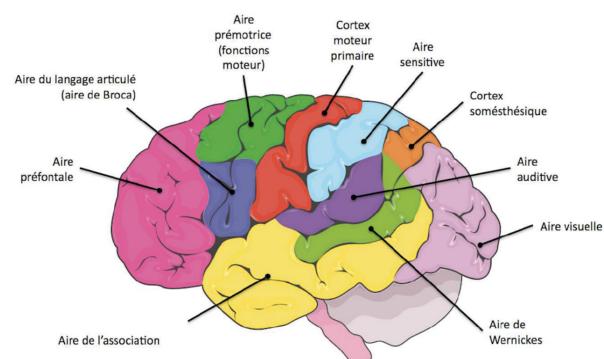


FIGURE 1.2 – Encéphale humain (« Noyaux gris centraux », s. d.).

le “cortex cérébral”. Il présente plusieurs circonvolutions qui augmentent considérablement sa surface. Le cortex cérébral est divisé en régions fonctionnelles que nous appelons “aires” (voir Figure 1.3b). Le travail de Dr. Broca sur le cas de M. Leborgne sont largement reconnus comme l’origine de cette division (FODOR, 1983).



(a) Lobes du cerveau (ART, 2013)



(b) Aires du cortex cérébral (JDIFOOL, 2006)

FIGURE 1.3 – Division morphologique et fonctionnelle du cerveau.

En particulier, la région du cerveau de M. Leborgne où Dr. Broca observe la lésion, correspond à l’aire qui porte son nom (aire de Broca). Ce dernier et celui de Wernicke jouent un rôle pivot pour l’aphasie (HALLOWELL, 2017).

### 1.1.2 Classification des syndromes aphasique

La définition que nous avons donnée de l'aphasie s'applique à une multitude de troubles dissimilaires en cause (région touchée du cerveau) et effet (conséquences pour la communication) (HALLOWELL, 2017). On parle de *syndromes*. La table 1.1 présente une classification des syndromes aphasiques classiques. En particulier, notre sujet d'intérêt, l'aphasie

Syndrome Aphasiq	Expressive / Récepti	Localisation de la Lésion	Effet sur la Compréhension	Effet sur l'Expres
Aphasie de Wernicke	Récepti	Aire de Wernicke (Brodmann 22 <sup>1</sup> )	Modéré à sévère	Modéré à sévère
Aphasie de Broca	Expressive	Aire de Broca (Brodmann 44, 45)	Léger à modéré	Modéré à sévère
Aphasie de Conduction	Les deux	Brodmann 40	Léger à modéré	Léger à modéré
Aphasie Globale	Expressive	Large, touche à plusieurs régions	Sévère	Sévère
Aphasie Transcorticale Sensorielle	Récepti	Brodmann 37, 39	Modéré à sévère	Modéré à sévère
Aphasie Transcorticale Motrice	Expressive	Brodmann 6, 8–10, 46	Léger à modéré	Léger à modéré
Aphasie Transcorticale Mixte	Expressive	Lobe Frontal inférieur	Léger à modéré	Léger à modéré

TABLE 1.1 – Classification des syndromes aphasiques classiques (HALLOWELL, 2017)

de Broca, est une aphasie expressive. Elle affecte surtout la production du langage. Sa compréhension est généralement préservée. De ce fait, les personnes atteintes de l'aphasie de Broca sont conscientes de leur handicap (CHAPEY, 2008).

### 1.1.3 Causes, prévalence et incidence de l'aphasie

Les AVC sont la première cause d'aphasie (HALLOWELL, 2017). En effet, 30% des individus atteints d'un AVC développent une aphasie (FLOWERS et al., 2016). Parmi ces individus, 43% ont une aphasie de Broca (CNSA, 2015). Étant donné que 13 millions de personnes sont touchées par un AVC chaque année (SMAÏLI et al., 2022), cela représente environ 3.9 millions de cas d'aphasie par an (une incidence d'un peu moins de 0.05%). Inversement, 75% des cas d'aphasie sont causés par un AVC (CNSA, 2015).

Il est difficile d'estimer l'incidence et la prévalence globales de l'aphasie. Ceci est due au manque de données dans la majorité des pays du monde. Cependant, le peu de données disponibles donnent des valeurs consistantes avec le 0.05% estimé ci-dessus. Selon l'association nationale de l'aphasie (« National Aphasia Association », s. d.), 2 millions Américains en souffrent, une prévalence de 0.6%. En France, ce chiffre est de l'ordre de

1. BRODMANN, 2007.

300000 cas, 30000 desquels sont nouveaux (CNSA, 2015). Ceci donne une prévalence de 0.44% et un taux d'incidence 0.044%.

L'âge est un facteur de risque très important pour les AVC, il l'est donc également pour l'aphasie. En effet, l'âge moyen des individus Français atteints de l'aphasie est de 73 ans. 75% parmi eux sont âgés de plus de 65 ans dont 25% dépassent les 80 ans (CNSA, 2015).

#### 1.1.4 Effet de l'aphasie de Broca sur la qualité de vie

Il n'est pas surprenant que l'aphasie ait un impact négatif sur la qualité de vie des individus qui en souffrent. En effet, les individus atteints d'aphasie ont une moins bonne qualité de vie que les individus en bonne santé selon les critères WHOQOL-BREF<sup>2</sup> et WPI<sup>3</sup>(voir Table 1.2, ROSS et WERTZ, 2010).

Measure	Mean	Range	SD	Mean difference	95% C. I. <sup>a</sup> of difference	t(34)
<i>WHOQOL-BREF Transformed total</i>						
NBI participants	108.44	94.00–125.00	10.02			
Aphasic participants	96.11	68.00–124.00	14.05	12.23	4.07–20.60	3.03**
<i>WHOQOL-BREF overall QOL and general health rating</i>						
NBI participants	8.44	6.00–10.00	1.58			
Aphasic participants	7.22	4.00–10.00	1.52	1.22	0.17–2.27	2.37*
<i>PWI total</i>						
NBI participants	36.33	28.00–42.00	3.40			
Aphasic participants	31.72	22.00–41.00	5.90	4.61	1.35–2.87	2.87**

TABLE 1.2 – Comparaison de la qualité de vie chez les individus saints et ceux qui souffrent d'une aphasia (ROSS & WERTZ, 2010).

En particulier, l'aphasie de Broca diminue mesurément la qualité de vie des individus qu'elle affecte dans toute tâche qui nécessite la communication (PALLAVI et al., 2018). La Table 1.4 le montre bien pour les trois exemples d'activités sociales, confiances en soi et capacité à réaliser ses responsabilités quotidiennes.

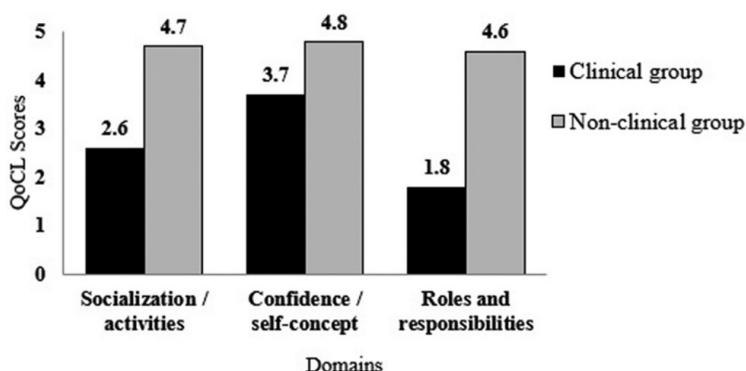


FIGURE 1.4 – Comparaison de la qualité de vie de communication chez les individus saints et ceux qui souffrent de l'aphasie de Broca (PALLAVI et al., 2018).

2. <https://www.who.int/publications-detail-redirect/WHO-HIS-HSI-Rev.2012.03>  
3. <https://www.acqol.com.au/instruments>

L'aphasie de Broca est également associée à une augmentation du risque d'hospitalisation et de décès (FLOWERS et al., 2016). Elle est aussi corrélée à un risque accru de maladies mentales, dépression et même tentative de suicide (COSTANZA et al., 2021 ; MORRISON, 2016). Ce dernier point est particulièrement grave en raison du fait que les personnes atteintes de l'aphasie de Broca sont généralement (1) âgées, (2) socialement isolées à cause de leur sentiment d'infériorité et (3) ont du mal à communiquer leur intention de suicide à cause de leur handicap. Il est donc urgent d'intervenir pour mitiger ces risques ainsi que la sévérité des effets qu'a l'aphasie.

## 1.2 Solution proposée

Il n'existe pas de traitement général de l'aphasie de Broca. L'intervention thérapeutique est donc adaptée à chaque patient (ACHARYA & WROTON, 2022). Un point commun à la plupart des thérapies, est l'utilisation d'exercices orthophoniques pour la rééducation de la parole. Des exemples de tels exercices sont la répétition de phrases, la description d'images et la narration de récits (da FONTOURA et al., 2012). En dépit d'être la méthode la plus efficace dont on dispose, cette approche est très chronophage étant donné que la majorité des patients ne bénéficient que de 1–3 séances par semaine (da FONTOURA et al., 2012). Elle est également coûteuse, avec un coût moyen dans les milliers de dollars par patient (JACOBS & ELLIS, 2021 ; LIU et al., 2021), ce qui la rend très inaccessible. Ces lacunes sont inacceptables dans le contexte des effets dévastateurs de l'aphasie de Broca que nous avons décrits précédemment.

### 1.2.1 Automatisation de la rééducation de la parole

Une façon d'augmenter l'accessibilité de la rééducation de la parole — et donc de mitiger les effets de l'aphasie de Broca — est de l'automatiser. Un système automatique de rééducation n'a aucune contrainte temporelle. Le nombre de séances par semaine n'est limité que par la disponibilité du patient, plutôt que par le nombre d'orthophonistes disponibles. Il élimine également le deuxième obstacle à l'accessibilité en ramenant le coût du traitement à celui du matériel nécessaire.

Dans ce travail, nous proposons la création d'un système informatique pour la correction de la parole produite par une personne atteinte de l'aphasie de Broca. Il s'agit d'une tâche assez commune dans le traitement orthophonique dispensé à ces patients (ACHARYA & WROTON, 2022 ; da FONTOURA et al., 2012). Le système doit être capable de transformer l'audio  $S_A$  (pour son aphasique) de la parole produite par le patient en l'audio  $S_C$  (pour son corrigé) d'une parole correcte (voir la Figure 1.5).

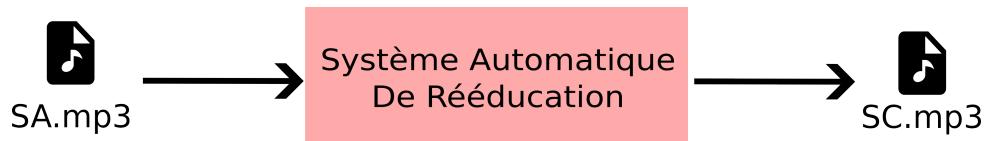


FIGURE 1.5 – Diagramme de fonctionnement du système proposé.

### 1.2.2 Structure proposée pour la solution

Au lieu de mapper l'audio erroné directement à l'audio corriger, nous proposons un système qui enchaîne trois transformations. La première transforme  $S_A$  en une représentation textuelle  $T_A$ . La deuxième produit un texte corrigé  $T_C$  à partir de  $T_A$ . La dernière étape est de produire  $S_C$  à partir de  $T_C$  (voir la Figure 1.6).

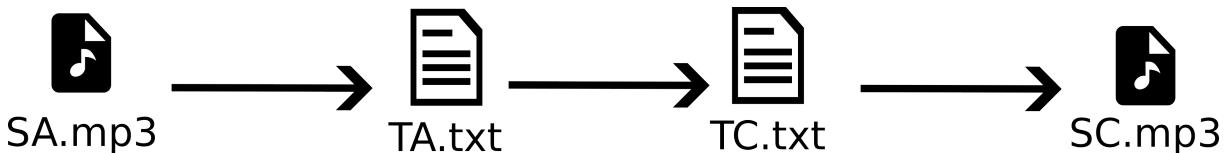


FIGURE 1.6 – Étapes intermédiaires du système proposé.

L'étape finale de cette procédure est la plus simple et peut être considérée comme résolue. La tâche en question est tout simplement de produire un audio à partir d'un texte. Il s'agit de synthèse vocale. Plusieurs solutions existent pour cette tâche, y compris des solutions open-source (TAN et al., 2022). Nous nous concentrerons donc sur les deux premières étapes.

La première partie de notre système réalise une transcription automatique de l'audio. C'est une tâche de reconnaissance automatique de la parole (ASR, de l'anglais : automatic speech recognition), avec la complexité supplémentaire que la parole en question ne suit pas les lois usuelles du langage.

La seconde partie est sans doute la plus complexe. C'est la partie centrale de notre système qui réalise directement la correction. Pour ce projet, nous avons choisi de traiter le problème comme un problème de traduction automatique (MT, de l'anglais : machine translation). Autrement dit, le texte erroné et le texte corrigé sont considérés comme s'ils appartiennent à deux langues différentes, entre lesquelles nous voulons réaliser une correspondance.

Ces deux étapes sont deux exemples de tâches de traitement automatique du langage (NLP, de l'anglais : natural language processing). Plus précisément, elles sont des tâches de modélisation séquence-à-séquence (S2S).

Dans le reste de ce document, nous détaillerons l'état de l'art de ces deux tâches. Le chapitre suivant présente en détail le problème général de modélisation S2S, ainsi que les différentes approches de apprentissage profond (DL, de l'anglais : deep learning) qui ont été proposées dans la littérature pour le résoudre. Cela a pour but de choisir une approche de DL pour implémenter les deux premières composantes de notre système.

Celui qui suit aborde plus spécifiquement les problèmes d'ASR et de MT dans le contexte où elles peuvent être appliquées dans notre système. Nous y présentons une revue de la littérature pour le cas particulier des tâches d'intérêt. Dans le cas où elles existent, nous présentons des cas d'applications similaires à notre problème.

## 1.3 Conclusion

Dans ce chapitre, nous avons introduit l'aphasie en général et l'aphasie de Broca en particulier comme troubles de langage. Nous les avons mis dans le contexte historique et scientifique en présentant leurs causes, portée et conséquences pour les patients qui en souffrent.

Après cela, nous avons discuté les traitements typiquement déployés contre l'aphasie de Broca. En particulier, nous avons discuté le manque d'inaccessibilité et de scalabilité du traitement orthophonique. Un problème que nous avons jugé très grave à cause des conséquences dévastatrices de l'aphasie de Broca.

Pour y remédier, nous avons proposé la création d'une solution informatique. Nous avons décrit son fonctionnement et proposé une architecture possibles pour sa mise en œuvre. Dans les chapitres suivants, nous allons explorer les méthodes possibles pour implémenter notre solution.

# Chapitre 2

## Apprentissage séquence-à-séquence

Les modèles S2S sont une famille d’algorithmes d’apprentissage automatique (ML, de l’anglais : machine learning) dont l’entrée et la sortie sont des séquences (MARTINS, 2018). Plusieurs tâches d’ML, notamment en NLP, peuvent être formulées comme tâches d’apprentissage S2S. Parmi ces tâches, nous citons : la création de chatbots, la réponse aux questions et –crucialement pour ce travail– la MT et la ASR (FATHI, 2021).

Dans ce chapitre, nous commençons par formuler le problème de modélisation de séquences. En suite, nous présentons les architectures neuronales les plus utilisées pour cette tâche. Enfin, nous terminons avec une étude comparative de celles-ci.

### 2.1 Énoncé du problème

Formellement, le problème de modélisation S2S est celui de calculer une fonction partielle  $f : X^* \rightarrow Y^*$ , où :

- $X$  est un ensemble dit d’entrées.
- $Y$  est un ensemble dit de sorties.
- Pour un ensemble  $A$ ,  $A^* = \bigcup_{n \in \mathbb{N}} A^n$  est l’ensemble de suites de longueur finie d’éléments de  $A$ .

$f$  prend donc un  $x = (x_1, x_2, \dots, x_n) \in X^n$  et renvoie un  $y = (y_1, y_2, \dots, y_m) \in Y^m$ . Dans le cas général,  $n \neq m$  et aucune hypothèse d’alignement n’est supposée. Il est souvent de prendre  $X = \mathbb{R}^{d_e}$  et  $Y = \mathbb{R}^{d_s}$  avec  $d_e, d_s \in \mathbb{N}$ . Dans ce cas,  $x \in \mathbb{R}^{d_e \times n}$  et  $y \in \mathbb{R}^{d_s \times m}$ . Les indices peuvent représenter une succession temporelle ou un ordre plus abstrait comme celui des mots dans une phrase (MARTINS, 2018).

La majorité des outils mathématiques historiquement utilisés pour ce problème viennent de la théorie du traitement de signal numérique. Cependant, l’approche actuellement dominante et celle qui a fait preuve de plus de succès, est de les combiner avec les réseaux de neurones.

## 2.2 Perceptrons multicouches

Les réseaux de neurones profonds sont parmi les modèles les plus expressifs en ML. Leur succès pratique est incomparable aux modèles qui les ont précédés, que se soit en termes de qualité des résultats ou de variétés de domaines d'application (SEBASTIAN & MIRJALILI, 2017). De plus, grâce aux théorèmes dits d'approximation universelle, ce succès empirique est formellement assuré (CALIN, 2020).

### 2.2.1 Généralités

Dans cette section, nous introduisons les perceptrons multicouches (MLP, de l'anglais : multi-layer perceptron), l'architecture neuronale la plus simple et la plus utilisée. Il s'agit d'une simple composition de couches affines avec des activations non affines (voir Définition 1).

**Définition 1** (MLP, MUKHERJEE, 2021).

Soient  $k, w_0, w_1, \dots, w_{k+1} \in \mathbb{N}$ , un réseau de neurones feed-forward de profondeur  $k + 1$ , à  $w_0$  entrées et  $w_{k+1}$  sorties, est défini par une fonction :

$$\begin{cases} \mathbb{R}^{w_0} \rightarrow \mathbb{R}^{w_{k+1}} \\ x \mapsto \varphi_{k+1} \circ A_{k+1} \circ \varphi_k \circ A_k \circ \dots \circ \varphi_1 \circ A_1(x) \end{cases} \quad (2.1)$$

Où les  $A_i$  sont des fonctions affines  $\mathbb{R}^{w_{i-1}} \rightarrow \mathbb{R}^{w_i}$  et les  $\varphi_i$  sont des fonctions quelconques, typiquement non affines  $\mathbb{R}^{w_i} \rightarrow \mathbb{R}^{w_i}$ , dites *d'activations*. La fonction  $\varphi_i \circ A_i$  est appelée la  $i^{\text{eme}}$  couche du réseau.

À un tel réseau de neurones, on peut associer un graphe orienté acyclique qu'on appelle son “architecture sous-jacente” (KEARNS & VAZIRANI, 1994). La Figure 2.1 illustre l'architecture d'un MLP de profondeur 4 avec  $(w_0, w_1, w_2, w_3, w_4) = (4, 5, 7, 5, 4)$ .

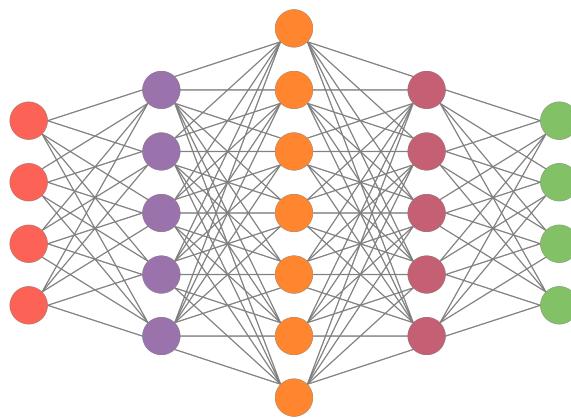


FIGURE 2.1 – Architecture sous-jacente d'un MLP de profondeur 4.

Deux MLP peuvent avoir la même architecture sous-jacente, en effet, cette dernière ne dépend que des dimensions de leurs couches respectives. De ce fait, une méthode de trouver pour une architecture et une fonction cible données le meilleur MLP est nécessaire.

Pour ce faire, nous exploitons le fait que les  $A_i$  soient des applications affines sur des espaces de dimensions finies. Nous pouvons donc les écrire comme combinaisons de produits matriciels et de translations. Le problème se réduit donc à régler<sup>1</sup> les paramètres des matrices en question. Cela nécessite une façon de quantifier la qualité d'approximation d'une fonction  $f$  par une autre  $\hat{f}$ . L'analyse fonctionnelle nous en donne plusieurs, les équations (2.2) et (2.3) sont deux exemples récurrents de fonctions dites *de perte*.

$$L_1(f, \hat{f}) = \|f - \hat{f}\|_1 = \int |f - \hat{f}| \quad (2.2)$$

$$L_2(f, \hat{f}) = \|f - \hat{f}\|_2^2 = \int |f - \hat{f}|^2 \quad (2.3)$$

Ayant fixé une fonction de perte  $L$ , l'entraînement revient à un problème d'optimisation comme représenté par l'équation 2.4.

$$f^* = \underset{\hat{f}}{\operatorname{argmin}} L(f, \hat{f}) \quad (2.4)$$

Dans le cas particulier où  $L$  est différentiable, l'algorithme du gradient peut être utilisé pour trouver un minimum local. Les gradients sont calculés en utilisant une méthode de dérivation automatique comme la rétro-propagation.

## 2.2.2 Application à la modélisation de séquence

Les réseaux de neurones opèrent sur des vecteurs. À fin de les utiliser dans le contexte de la modélisation S2S, il faut donc utiliser une représentation vectorielle des entrées. Une telle représentation s'appelle un *plongement* (embedding en anglais). Le plongement peut-être appris ou prédéfini.

Dans le cas des MLP, la séquence d'entrée est d'abord décomposée en sous-séquences. Ensuite, les plongements de ces sous-séquences sont traités un par un par le réseau de neurones, ce qui produit une séquence de vecteurs en sortie. (voir le Bloc de code 2.1).

```

1 def mlp_sequence(input: Sequence):
2     """
3     @param input: The sequence to be processed
4     """
5     output = []
6     for element in input:
7         e = embedding(element)
8         output.append(e)
9     return output

```

Algorithme 2.1 – Passe d'un MLP

---

1. En ML, le terme “entraîner” est plutôt utilisé.

### 2.2.3 Avantages et inconvénients

Les MLP présentent deux avantages par rapport aux architectures discutées dans le reste de ce chapitre. Le premier est leur simplicité. Elle les rend plus simples à comprendre et à implémenter. Le deuxième est le fait qu'ils traitent indépendamment les sous-séquences. Cela rend très facile la tâche de les paralléliser, et par conséquent, il accélère considérablement leur entraînement.

Cependant, ce dernier point pose un grand problème. Comme ils traitent indépendamment les blocs de la séquence, les MLP ne peuvent pas modéliser les dépendances inter-bloc. Par conséquent, leur performance sur les séquences composées de plusieurs blocs est très médiocre. La solution de ce problème est d'augmenter la taille du bloc (est donc aussi la dimension d'entrée). Or, cela suppose un alignement par blocs entre les deux séquences. Une hypothèse invalide selon la Section 2.1.

## 2.3 Architecture encodeur–décodeur

Les lacunes des MLP sont en grande partie due au traitement séparé des parties des séquences. Dans la production de l'élément (ou bloc) courant de la sortie, un MLP se base uniquement sur l'élément (ou bloc) correspondant de l'entrée. L'équation 2.5 l'illustre pour une entrée  $x = (x_1, x_2, \dots, x_n)$  et une sortie  $y = (y_1, y_2, \dots, y_m)$ .

$$y_j = f(x_i, x_{i+1}, \dots, x_{i+\ell}) \quad 1 \leq j \leq m \quad (2.5)$$

En plus de l'hypothèse implicite de l'existence d'une telle correspondance, cela suppose que les éléments d'une séquence sont complètement indépendants l'un de l'autre. Cette dernière hypothèse n'est presque jamais vérifiée.

Une façon naturelle de combler ces lacunes est d'abandonner le traitement par bloc de l'entrée. Tout élément de la séquence de sortie est considéré comme fonction de la séquence d'entrée toute entière. L'équation 2.6 montre cette approche sur l'exemple précédent.

$$y_j = f(x) = f(x_1, x_2, \dots, x_n) \quad 1 \leq j \leq m \quad (2.6)$$

L'architecture encodeur–décodeur y parvient en combinant deux modules : un encodeur et un décodeur. L'encodeur consomme la suite d'entrée et produit un vecteur de dimension fixe qui la représente.<sup>2</sup> Le décodeur consomme ce vecteur et produit la sortie (voir Figure 2.2). L'équation 2.7 le montre sur le même exemple.

$$\begin{aligned} c &= \text{encoder}(x) \\ y &= \text{decoder}(c) \end{aligned} \quad (2.7)$$

L'équation 2.7 ne dépend pas du fonctionnement interne de l'encodeur et du décodeur. Les deux modules peuvent avoir deux architectures quelconques, qui peuvent ou non être les mêmes (YANG et al., 2020). Dans le reste de ce chapitre, nous examinons les architectures communes en apprentissage S2S.

---

2. Ce vecteur est appelé un vecteur de contexte, vecteur de pensée ou encore un encodage.

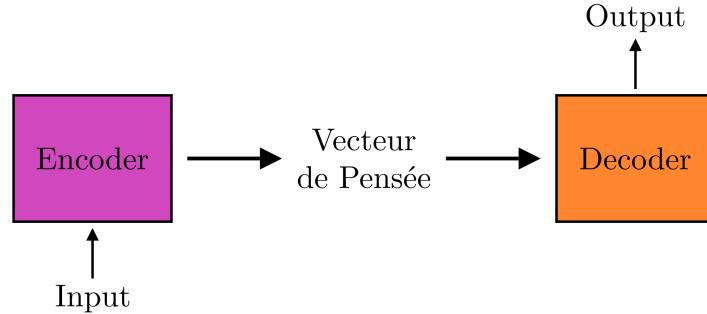


FIGURE 2.2 – Architecture encodeur-décodeur

## 2.4 Réseaux de neurones récurrents

L'un des principaux défauts que nous avons observés avec les MLP et qui nous ont poussés à introduire l'architecture encodeur-décodeur, est leur incapacité de représenter la dépendance entre les éléments d'une séquence. Une incapacité qui résulte de leur traitement indépendant des éléments.

Les réseaux de neurones récurrents (RNN, de l'anglais : recurrent neural network) tentent à résoudre ce problème en utilisant un état interne persistant. Chaque élément de la séquence modifie cet état lors de son traitement. Cela permet aux premiers éléments d'affecter le traitement des éléments qui les suivent, ainsi permettant à l'information de se propager vers le futur, ce qui donne lieu à une *mémoire*. Pour implémenter ce type de

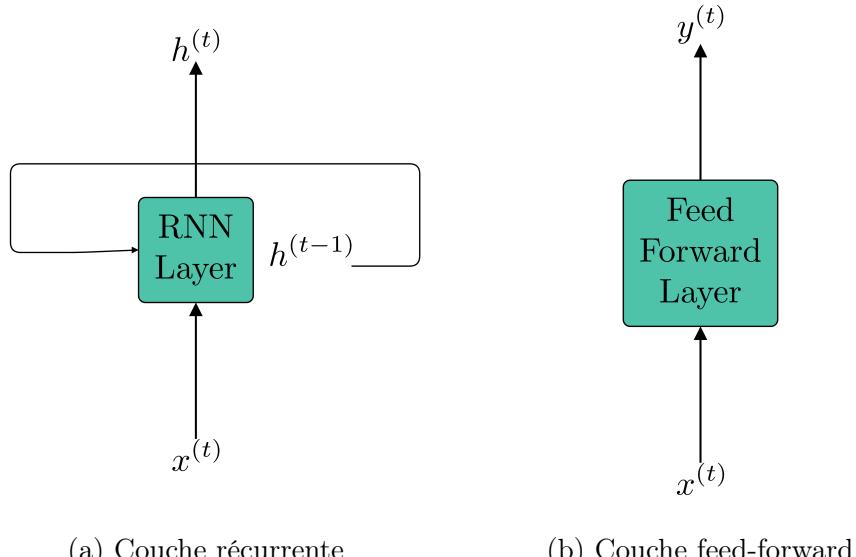


FIGURE 2.3 – RNN v.s FFN

comportement, l'état d'un RNN est décalé d'une unité est réinjecté dans l'entrée (voir Figure 2.3a). Cela est très différent des MLP qui n'ont pas de boucle de rétroaction, il s'agit de réseau de neurones feed-forwards (FFN, de l'anglais : feed-forward network

longplural) (voir Figure 2.3b). Par conséquent, ils n'ont pas d'état ni de mémoire (FATHI, 2021).

### 2.4.1 Réseaux de neurones récurrents simples

L'RNN le plus simple à imaginer se réduit à une combinaison affine de l'état passé et de l'entrée. Il permet de l'information sur le passé de se propager sans contrôle particulier. Mathématiquement, une couche d'un tel RNN prend la forme suivante

$$h^{(t)} = \varphi(Uh^{(t-1)} + Wx^{(t)} + b) \quad 1 \leq t \leq n \quad (2.8)$$

où  $t$  est le temps<sup>3</sup>,  $x^{(t)}, h^{(t)}$  sont respectivement l'entrée et la sortie à l'instant  $t$ ,  $n$  est la longueur de la séquence et  $\varphi$  est la fonction d'activation (FATHI, 2021). Dans le cas où  $\varphi$  est l'identité, la transformée en  $z$  de l'équation 2.8 est donnée par

$$H(z) = z(zI - U)^{-1}(WX(Z) + b) \quad (2.9)$$

il s'agit donc d'un système à réponse impulsionnelle infinie (FATHI, 2021).

#### Dépliement temporel et encodeur–décodeur récurrent

Le traitement d'une séquence  $x$  par un RNN ( $\mathcal{R}$ ), est équivalent à son traitement par un FFN ( $\mathcal{F}$ ) dont la profondeur est égale à la longueur de  $x$ .  $\mathcal{F}$  est donc appelé *dépliement temporel* de  $\mathcal{R}$  pour  $x$ . La Figure 2.4 montre le dépliement temporel de la Figure 2.3a pour une séquence de longueur 4 (LECUN et al., 2015).

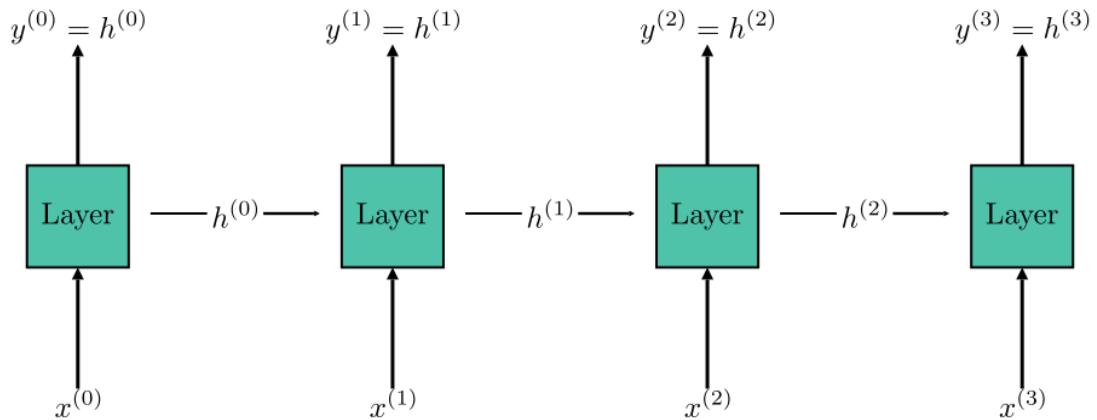


FIGURE 2.4 – Dépliement temporel d'un RNN sur une entrée de longueur 4.

Chaque état caché  $h^{(i)}$  contient de l'information sur tous les  $x^{(j)}$ ,  $j \leq i$ . En particulier,  $h^{(n)}$  contient de l'information sur toute la séquence  $x$ . Un encodeur récurrent peut donc retourner son dernier état caché comme vecteur d'encodage (YANG et al., 2020).

---

3. Il peut être continu ou discret, réel ou abstrait.

De sa part, un décodeur récurrent peut conditionner sur l'encodage et passer ses états cachés à une couche supplémentaire qui les interprète comme plongements des éléments de la sortie  $y$  (FATHI, 2021). Un tel décodeur n'a pas besoin d'entrée séquentielle (voir Figure 2.5)

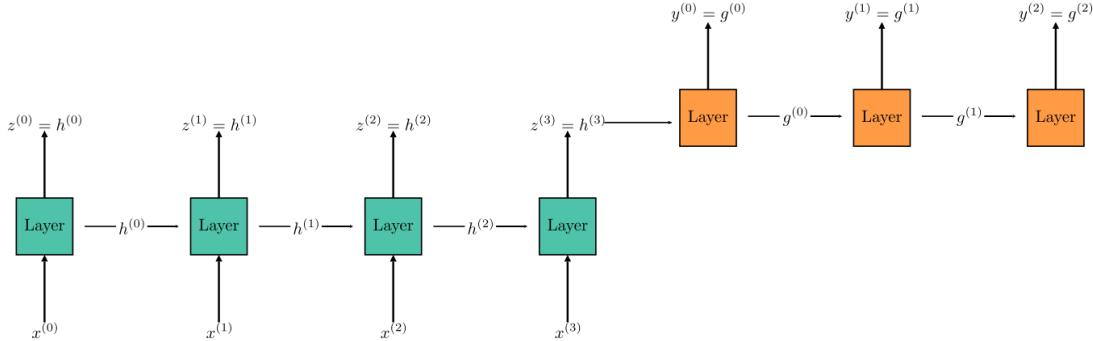


FIGURE 2.5 – Dépliement temporel d'un encodeur–décodeur récurrent sur une entrée de longueur 4 et une sortie de longueur 3.

### Rétro-propagation dans le temps et mémoire à court terme

L'entraînement d'un RNN sur un exemple se fait en le dépliant, puis en entraînant le FFN qui résulte par rétro-propagation. L'algorithme résultant s'appel la rétro-propagation dans le temps (BPTT, de l'anglais : back-propagation through time). Cela est problématique, car la taille de l'entrée n'est théoriquement pas bornées. Par conséquent, la profondeur effective d'un RNN ne l'est pas non plus (FATHI, 2021).

Or, dans l'entraînement d'un réseau de neurones trop profond, les modules des gradients peuvent atteindre des valeurs trop grandes ou trop petites. Il s'agit respectivement des problèmes de “l’explosion du gradient” et de “la disparition du gradient” (BASODI et al., 2020) Une conséquence de ce phénomène est que les RNN simples ont du mal avec les entrées pour lesquelles le dépliement temporel est profond, (i.e les longues entrées). Cette incapacité à modéliser les corrélations à long terme est appelée “mémoire à court terme” (BENGIO et al., 1994 ; INFORMATIK et al., 2003).

### 2.4.2 Portes, gated recurrent unit et long short-term memory

Pour une grande partie des problèmes d'apprentissage S2S, les séquences peuvent être très longues. Le mémoire à court terme constitue donc un véritable obstacle pour l'utilisation des RNN simples en pratique. Une approche de le contourner qui a eu un énorme succès expérimental, est l'introduction d'un mécanisme de contrôle sur la boucle de rétroaction (voir Figure 2.6). Ce mécanisme est généralement implémenté avec des *portes*, des unités entraînables qui peuvent réguler le flux d'information dans la couche récurrente. On parle alors d'*RNN à portes*. Dans cette section, nous explorons les deux

variants les plus utilisés d'RNN à portes : le gated recurrent unit (GRU) et le long short-term memory (LSTM).

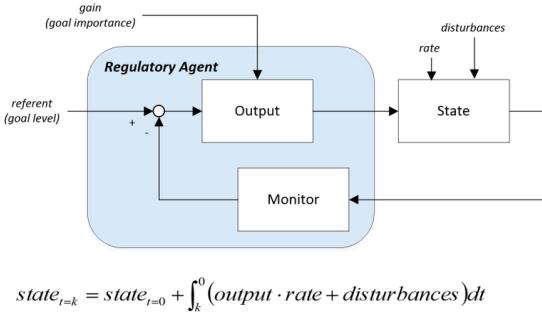


FIGURE 2.6 – Forme générale d'un RNN à portes.

### Gated recurrent unit

Le GRU, introduit par (CHO et al., 2014), est une architecture récurrente à portes très simple (voir Figure 2.7). Elle utilise deux portes. La première est la porte de réinitialisation (( $r$  dans la figure 2.7)). Elle détermine le point auquel l'information sur le passé peut se propager (quand  $r = 0$ , pas de propagation et quand  $r = 1$ , propagation totale). Sa sortie s'appelle l'état candidat ( $\tilde{h}$  dans la figure). La deuxième est la porte de mise à jour ( $z$  dans la figure). Elle détermine les contributions respectives de l'état candidat et l'état passé (si  $z = 0$ , seul l'état candidat contribue et si  $z = 1$ , seule l'état courant contribue). Sa sortie est la sortie globale du GRU (CHO et al., 2014).

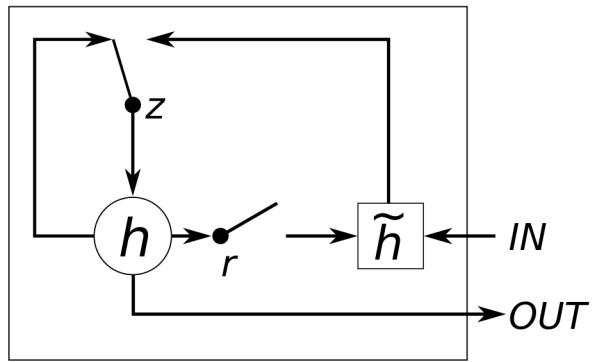


FIGURE 2.7 – Architecture interne d'un GRU (CHUNG et al., 2014, Fig. 1b)

Le fonctionnement des portes d'un GRU est simple. Leurs valeurs sont calculées à partir de l'entrée et de l'état courants par les équations 2.10 et 2.11, où  $\sigma$  est la fonction

*sigmoïde*<sup>4</sup> et  $\odot$  et le produit d’Hadamard<sup>5</sup>.

$$z^{(t)} = \sigma(W_z x^{(t)} + U_z h^{(t-1)} + b_z) \quad (2.10)$$

$$r^{(t)} = \sigma(W_r x^{(t)} + U_r h^{(t-1)} + b_r) \quad (2.11)$$

$$\tilde{h}^{(t)} = \varphi(W_h x^{(t)} + U_h (r^{(t)} \odot h^{(t-1)}) + b_h) \quad (2.12)$$

$$h^{(t)} = z^{(t)} \odot h^{(t-1)} + (1 - z^{(t)}) \odot \tilde{h}^{(t)} \quad (2.13)$$

L’état candidat est calculé à partir de l’entrée et l’état pondéré par la porte de réinitialisation par l’équation 2.12, où  $\varphi$  est la fonction d’activation. Finalement, l’état futur (la sortie) est la moyenne pondérée par  $z$  de l’état courant et l’état candidat (CHO et al., 2014). Notons que le GRU devient un RNN simple si les portes de réinitialisation et de mise à jour sont respectivement fixés à 1 et 0 (FATHI, 2021).

### Long short-term memory

Il s’agit de l’une des premières architectures récurrentes à protes (CHUNG et al., 2014). Elle a été introduite par (HOCHREITER & SCHMIDHUBER, 1997). Un LSTM implémente trois portes : une porte d’entrés ( $i$ ), une porte d’oubolie ( $f$ ) et une porte de sortie ( $o$ ) (voir Figure 2.8).

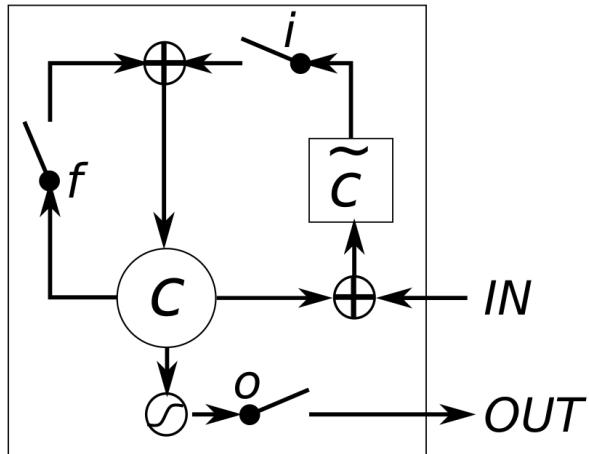


FIGURE 2.8 – Architecture interne d’un LSTM (CHUNG et al., 2014, Fig. 1a)

Le fonctionnement des portes est similaire à celui du GRU. Les équations 2.14–2.19 le montrent en détails<sup>6</sup> (HOCHREITER & SCHMIDHUBER, 1997).

4.  $\sigma : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto \frac{1}{1+e^{-x}}$

5. Pour  $u, v \in \mathbb{R}^n$ ,  $u \odot v \in \mathbb{R}^n$  et  $(u \odot v)_i = u_i v_i$

6.  $\tanh : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$$f^{(t)} = \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) \quad (2.14)$$

$$i^{(t)} = \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) \quad (2.15)$$

$$o^{(t)} = \sigma(W_o x^{(t)} + U_o h^{(t-1)} + b_o) \quad (2.16)$$

$$\tilde{c}^{(t)} = \tanh(W_c x^{(t)} + U_c h^{(t-1)} + b_c) \quad (2.17)$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \quad (2.18)$$

$$h^{(t)} = o^{(t)} \odot \varphi(c^{(t)}) \quad (2.19)$$

## Évaluation des réseaux de neurones récurrents

Nous avons établi que les RNN simples souffrent du mémoire à court terme (BENGIO et al., 1994 ; PASCANU et al., s. d.). En utilisant les portes pour contrôler le flux d'information, les GRU et LSTM résolvent le problème au prix d'architectures plus complexes et par conséquent plus lourdes à entraîner. Ils ont des performances similaires est largement meilleures que celle des RNN (CHUNG et al., 2014).

Cependant, toutes les architectures récurrentes présentent un problème fondamental : elles fonctionnent séquentiellement. Bien que cela les rend naturellement mieux adaptées à la modélisation de séquences, il les rend aussi quasi impossibles à paralléliser pour exploiter des architectures parallèles (GPU). Par conséquent, l'entraînement des RNN est extrêmement lent (STAHLBERG, 2020).

## 2.5 Réseau de neurones à convolutions

En dépit d'être une architecture naturelle pour le traitement des séquences, les RNN sont trop lents pour la majorité des cas d'utilisation pratiques. Cela est dû à leur nature séquentielle qui à son tour, est due à leur utilisation de boucles de rétroaction (voir Section 2.4). Une architecture sans telles boucles (i.e une architecture d'FFN) est donc préférable.

Les réseaux de neurones à convolutions (CNN, convolutional neural network) sont une famille d'FFN typiquement utilisés en traitement d'images. Ils ont été introduits par (FUKUSHIMA, 1980) et popularisés par (LECUN et al., 1989 ; LECUN et al., 1998). Les CNN atteignent des performances comparables à celles des RNN sans mémoire explicite. Ils y parviennent en exploitant un outil mathématique appelé produit de *convolution* (CALIN, 2020).

### Principes mathématiques de fonctionnement

Le produit de convolution de deux signaux  $f$  et  $g$  est le signal  $f * g$  donné par

$$u \mapsto \int_{\mathbb{R}} f(u-t)g(t)dt \quad (2.20)$$

il s'agit d'une opération commune en probabilité, analyse fonctionnelle, traitement de signaux et traitement d'images (BARBE & LEDOUX, 2012; OPPENHEIM & SCHAFER, 2013). À partir d'elle, une autre opération appelée *l'inter-corrélation* est définie. L'inter-corrélation de  $f$  et  $g$  est notée  $f \star g$ . Elle est définie par

$$u \mapsto \int_{\mathbb{R}} f(t)g(t-u)dt = (f * g^-)(u) \quad (2.21)$$

ou  $g^- : t \mapsto g(-t)$ . Intuitivement, elle mesure la similarité entre les deux signaux en question. Les CNN utilisent l'inter-corrélation à la place des applications linéaires quelconques d'un MLP. Une couche de convolution unidimensionnelle calcul donc la fonction suivante

$$x \mapsto \varphi \left( b + \sum_{i=1}^n x \star w_i \right) \quad (2.22)$$

ou  $x$  est l'entrée et les vecteurs  $w_i$  sont les paramètres entraînables de la couche, aussi appelés ses *noyaux* ou *masques* (voir Figure 2.9). La sortie de cette couche est une combinaison de tous les éléments de la séquence d'entrée. En composant suffisamment de telles couches, un CNN apprend une représentation de l'entrée beaucoup plus structurée qu'un MLP. On parle parfois de représentation *hiérarchique*.

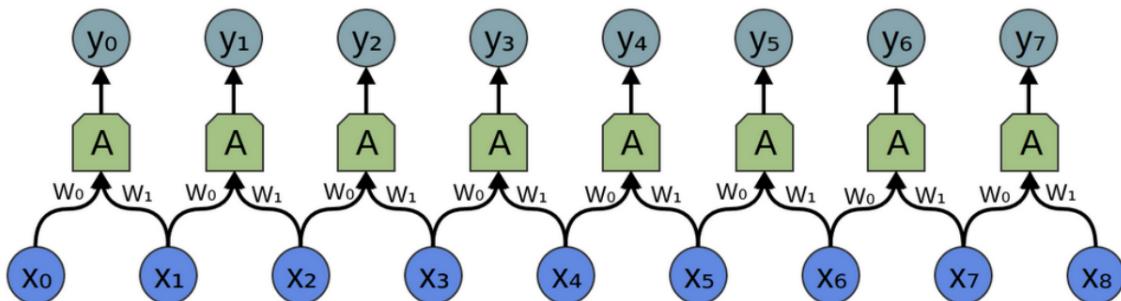


FIGURE 2.9 – Couche convulsive unidimensionnelle.

## Application à l'apprentissage S2S

Leur tendance naturelle à produire des représentations synthétiques des entrées, fait des CNN des encodeurs très puissants pour une grande variété de tâches, y compris des tâches de transduction de séquences. Plusieurs travaux ont combiné un encodeur convolutif avec un décodeur récurrent (voir Figure 2.10, KALCHBRENNER et BLUNSMON, 2013, Fig.3) (YANG et al., 2020).

Cependant, des architectures totalement convolutives pour l'apprentissage S2S existent également. ByteNet (voir Figure 2.11 KALCHBRENNER et al., 2017) et ConvS2S (voir Figure 2.12 KAMEOKA et al., 2020) sont deux exemples de telles architectures.

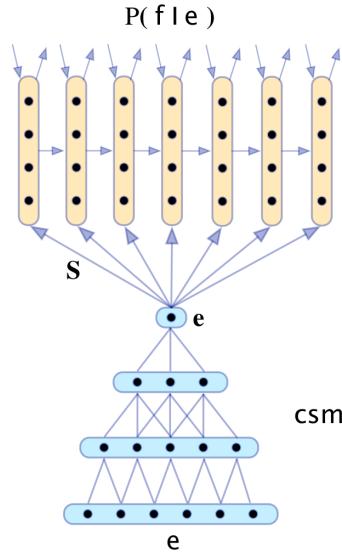


FIGURE 2.10 – Recurrent Continuous Translation Model

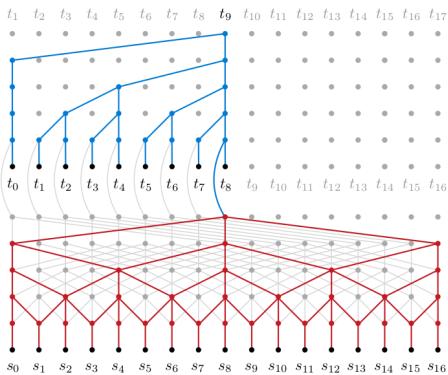


FIGURE 2.11 – Architecture de ByteNet.

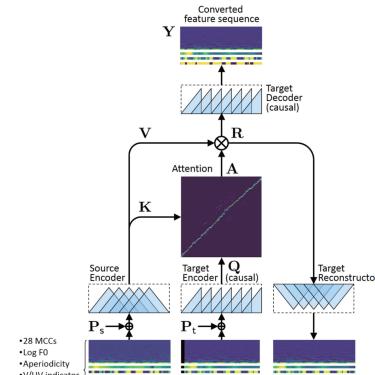


FIGURE 2.12 – Architecture de ConvS2S.

## Avantages et inconvénients

Le produit de convolution (et par conséquent, inter-corrélation) est une opération très facile à paralléliser. Il est donc possible d'exploiter des architectures matérielles parallèles (GPU) pour accélérer l'entraînement des CNN. À cet effet, les CNN sont beaucoup plus efficaces que les RNN pour les entrées longues. Ils peuvent être employés dans des cas pratiques de tailles raisonnables (LI et al., 2016).

Or, la nature locale du produit de convolution rend difficile aux CNN d'apprendre les corrélations globales<sup>7</sup>. En effet, pour représenter toutes les relations dans une séquence de longueur  $n$ , un CNN dont les masques sont de taille  $k < n$  doit avoir  $\log_k n$  couche (VASWANI et al., 2017). Le traitement de séquences très longues nécessite alors des CNN trop profonds, ce qui donne lieu au problèmes de dispartion et d'explosion des gradients.

7. Dépendances entre des couples d'éléments éloignés dans la séquence

## 2.6 Transformeurs

Les architectures discutées dans les sections 2.2 à 2.5 (notamment les encodeurs-décodeurs à base de CNN et RNN) forment la colonne vertébrale des modèles classiques d'apprentissage S2S (YANG et al., 2020). Cependant, leur mise en place est inhibée par des problèmes de performance (voir sections 2.4, 2.5 et 2.6.5).

Le transformeur (aussi appelé réseau de neurones auto-attentif) est une architecture performante pour le traitement de séquences (SHIM & SUNG, 2022). Introduite par (VASWANI et al., 2017), elle est basée sur le mécanisme d'attention (LAROCHELLE & HINTON, 2010). Une opération mathématique parallélisable à l'instar du produit de convolution, mais dont la complexité ne dépend pas de la distance dans les séquences.

### 2.6.1 Architecture générale

Le transformeur a une architecture encodeur-décodeur (voir Figure 2.13). L'encodeur et le décodeur ont des architectures très similaires, comprenant chacun une couche d'encodage positionnel, une couche d'auto-attention et un MLP. Le décodeur se distingue de l'encodeur en ce qu'il a une couche d'attention croisée entre l'auto-attention et l'MLP. Chacune des couches susmentionnées est suivie d'une couche de normalisation (BA et al., 2016) qui reçoit également une connexion directe à l'entrée de la couche<sup>8</sup>. L'encodeur et le décodeur peuvent être empilés pour former un transformeur profond. VASWANI et al., 2017 proposent 6 couches d'encodeur et 6 couches de décodeur.

Étant donné une séquence d'entrée  $x = (x_1, \dots, x_n)$ , l'encodeur calcule un vecteur de pensée  $z$  qu'il passe au décodeur. Le décodeur prédit le prochain élément  $y_i$  de la séquence de sortie à partir de  $z$  et les éléments précédents  $(y_1, \dots, y_{i-1})$ . Pour produire  $y_0$ , le décodeur commence avec une séquence de sortie vide.

### 2.6.2 Encodage positionnel

La première couche de l'encodeur aussi bien que du décodeur est une couche d'encodage positionnel. Elle permet de calculer une représentation vectorielle de la séquence en question. Elle est composée d'une couche de plongement et d'une couche d'encodage de la position.

La couche de plongement mappe chaque élément de la séquence à un vecteur de dimension fixe  $d$ , ainsi transformant une séquence de longueur  $n$  en une matrice de  $\mathbb{R}^{n \times d}$ . Elle peut avoir une architecture quelconque en fonction de la nature des données. Il peut s'agir d'une couche de plongement lexical dans le cas d'un texte, d'une couche de convolution dans le cas de l'audio ou d'une couche récurrente dans le cas d'une série chronologique.

Le transformeur est un invariant aux permutations<sup>9</sup>. La sortie de la couche de plonge-

8. On parle de connexion résiduelle ou de connexion de saut (skip connection) (HE et al., 2016)

9. Pour une séquence  $x = (x_1, \dots, x_n)$  et une permutation  $\pi \in S_n$ ,  $x$  et  $(x_{\pi(i)})_{1 \leq i \leq n}$  ont deux

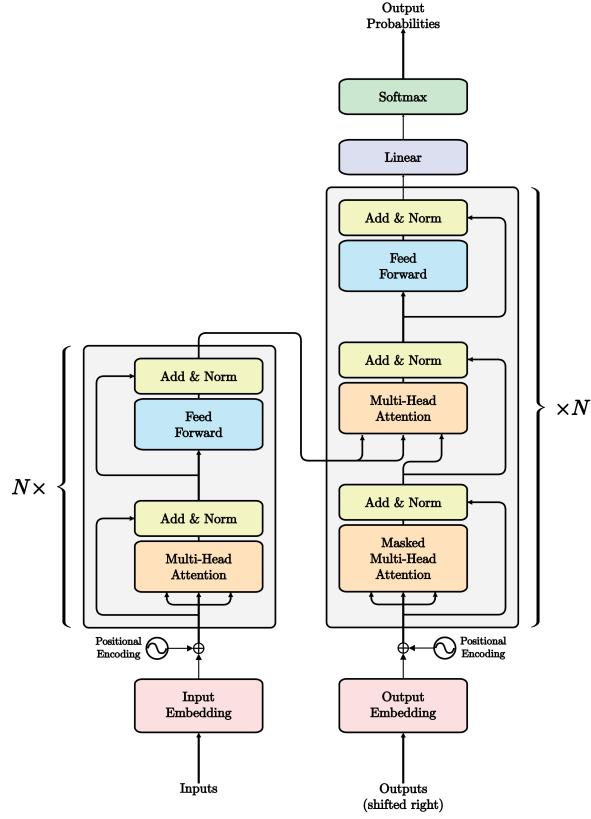


FIGURE 2.13 – Architecture de transformeur (VASWANI et al., 2017, Fig 1).

ment est donc insuffisante pour représenter la séquence. Il lui faut ajouter une information sur l'ordre des éléments. C'est ce que fait la couche d'encodage de la position. Les auteurs (VASWANI et al., 2017) proposent l'encodage suivant pour la position  $i$  :

$$\text{PE}_k(i) = \begin{cases} \sin\left(\frac{i}{r^{2k/d}}\right) & \text{si } k \text{ est pair} \\ \cos\left(\frac{i}{r^{2k/d}}\right) & \text{si } k \text{ est impair} \end{cases} \quad 1 \leq k \leq d \quad (2.23)$$

où  $d$  est la dimension de plongement et  $r$  est un hyperparamètre fixé à  $10^4$  par les auteurs. La figure 2.14 montre la matrice d'encodage des 256 premières positions avec  $r = 10^4$  et  $d = 512$ .

La représentation finale de la séquence est la somme de la sortie de la couche de plongement et celle de la couche d'encodage de la position. Une couche de plongement lexical peut être substituée à l'encodage donné par l'équation (2.23) avec des résultats similaires (VASWANI et al., 2017).

### 2.6.3 Couches attention

Le mécanisme d'attention est la partie centrale du transformeur qui réalise sa fonctionnalité. Tous les autres modules sont une forme de prétraitement de son entrée ou de encodages identiques.

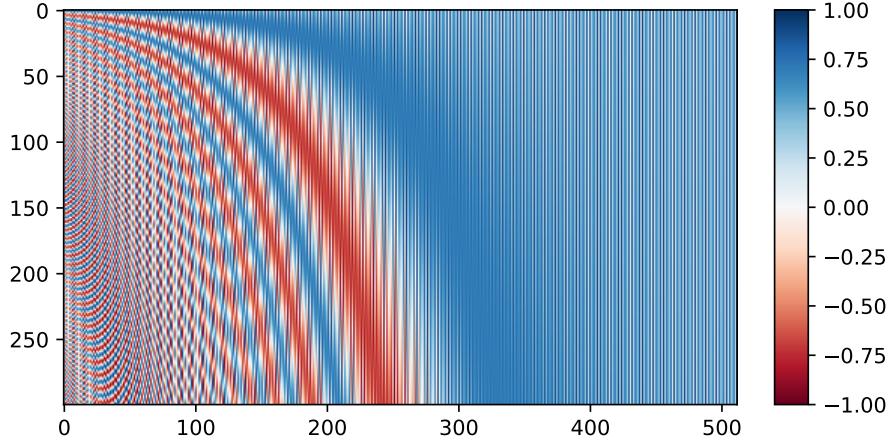


FIGURE 2.14 – Matrice d’encodage de la position pour  $r = 10^4$  et  $d = 512$ .

post-traitement de sa sortie. Chaque couche de l’encodeur implémente un module d’attention, tandis que chaque couche du décodeur en implémente deux.

Plusieurs types de mécanismes d’attention ont été proposés dans la littérature, notamment l’attention additive (BAHDANAU et al., 2016) et l’attention multiplicative (LUONG et al., 2015). Le transformeur utilise une variante de l’attention multiplicative appelée “*scaled dot-product attention*” (VASWANI et al., 2017). Elle opère sur trois matrices  $Q, K, V$ <sup>10</sup> de dimensions respectives  $d_Q \times l, d_K \times m, d_V \times n$ , qui représentent chacune l’encodage d’une séquence<sup>11</sup>. Le résultat est donné par l’équation (2.24).

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_K}} \right) V \quad (2.24)$$

Intuitivement, la scaled dot-product attention peut être vue comme une forme de recherche floue dans une base de données. Le terme  $QK^T$  est un produit scalaire entre les lignes de la requête  $Q$  et la clé  $K$ . Il représente la similarité entre les deux. La fonction softmax le normalise pour obtenir des poids positifs de somme 1. Enfin, le résultat est une somme des valeurs  $V$  pondérées par ces poids. C’est dans ce sens-là qu’il s’agit d’une recherche floue : les valeurs correspondantes aux clés qui répondent le mieux à la requête sont sélectionnées, mais au lieu de renvoyer discrètement la meilleure valeur, toutes les valeurs contribuent dans la mesure de leur pertinence (« CS480/680 Lecture 19 : Attention and Transformer Networks », 2019). La division par  $\sqrt{d_K}$  est une forme de régularisation. Elle permet d’éviter que le module de l’entrée du softmax devienne trop grand, ce qui rend son gradient trop petit ainsi ralentissant l’apprentissage<sup>12</sup> (VASWANI et al., 2017).

Les auteurs de (VASWANI et al., 2017) ont observé une amélioration de la performance en utilisant plusieurs modules (ou têtes) de scaled dot-product attention, projetant les entrées différemment avant de les passer à chaque tête. Les sorties des têtes sont concaténées

10. De la terminologie anglophone : “*Query*”, “*Key*”, “*Value*”, empruntée aux systèmes de recherche de l’information.

11.  $d_Q, d_K, d_V$  sont les dimensions de plongement et  $l, m, n$  sont les longueurs des séquences. Notez que l’équation (2.24) impose les contraintes  $d_Q = d_K$  et  $m = n$ , ce qui est vérifié pour les trois modules d’attention du transformeur.

12. Le choix de la valeur de  $\sqrt{d_K}$  n’est pas arbitraire. Sous l’hypothèse que les valeurs de  $Q$  et  $K$  sont centrées et réduites, les valeurs de  $QK^T$  sont centrées de variance  $d_K$ . La division par  $\sqrt{d_K}$  les ramène à une variance de 1 (VASWANI et al., 2017).

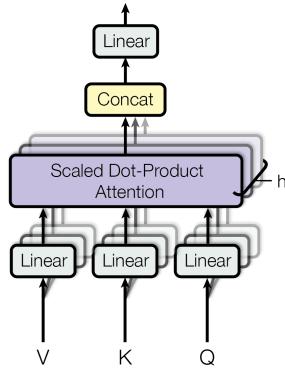


FIGURE 2.15 – Schéma d’une couche attention multi-tête (VASWANI et al., 2017, Fig 2).

et passées à une couche linéaire qui produit la sortie finale. Les projections sont réalisées par des couches linéaires entraînables et le nombre  $h$  de têtes est un hyperparamètre fixé à 8 par les auteurs(voir la Figure 2.15).

La couche attention de l’encodeur et la première couche attention du décodeur sont des couches d’*auto-attention* i.e leurs requêtes, clés et valeurs sont les mêmes. L’auto-attention du décodeur se distingue de celle de l’encodeur par la présence du *masque*. En produisant un élément de la séquence de sortie, le décodeur ne doit faire attention qu’aux éléments qui le précédent. Toutes les autres valeurs doivent avoir un poids nul. Pour ce faire, la matrice  $M$

$$M_{ij} = \begin{cases} 0 & \text{si } j > i \\ -\infty & \text{sinon} \end{cases} = \begin{bmatrix} -\infty & -\infty & \cdots & -\infty \\ 0 & -\infty & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\infty \end{bmatrix} \quad (2.25)$$

est ajoutée à l’entrée de la fonction softmax.

La deuxième couche attention du décodeur est une couche d’*attention croisée*. Les requêtes viennent de la sortie et les clés et valeurs viennent de l’entrée. Cette couche calcul les dépendances entre l’entrée et la sortie, tandis que les deux autres couches calculent les dépendances à l’intérieur de l’entrée et de la sortie. C’est cette couche qui permet au décodeur de conditionner sa sortie sur l’entrée.

## 2.6.4 Autres éléments de l’architecture

L’encodage positionnel et l’auto-attention sont les deux innovations clés du transformeur. Les autres éléments de l’architecture existent comme portes d’entrée et de sortie pour elles. Ces éléments sont les suivants :

- Les couches feed forward : Il s’agit d’MLP qui exploitent la représentation calculée par le mécanisme d’attention.
- Les couches de normalisation : Introduites par (BA et al., 2016), elles sont appliquées après chaque module. Elles assurent que leur sortie est centrée et réduite. Cela permet de stabiliser l’apprentissage.

- Les connexions résiduelles (HE et al., 2016) : Appliquées à chaque couche de normalisation, elles ont la forme LayerNorm ( $x + \text{Module}(x)$ ). Elles permettent aux gradients de se propager plus facilement.
- La couche de sortie : Dans le cas où les éléments de la sortie sont des variables continues (par exemple pour amplitudes d'un signal audio), une couche linéaire est appliquée à la sortie du dernier module. Dans le cas opposé (par exemple pour les mots d'une phrase), la fonction softmax est aussi appliquée pour donner la loi de probabilité de la sortie.

### 2.6.5 Analyse comparative de la performance

Dans le début de cette section, nous affirmons la supériorité du transformeur aux autres architectures S2S. Ce constat repose sur la comparaison du Tableau 2.1 (VASWANI et al., 2017). On y voit que le transformeur est le seul à avoir une longueur de chemin constante, c'est-à-dire qu'elle possède un majorant indépendant de la taille de l'entrée.

On note également que le transformeur exige un nombre d'opérations séquentielles constant, ce qui est le cas pour les autres architectures à l'exception des RNN. En effet, tous les réseaux feed-forward ont cette propriété.

Pour la complexité par couches, l'auto-attention est la plus efficace pour les séquences courtes (pour lesquelles  $n \ll d$ ). Ces séquences sont les plus fréquentes en pratique pour les dimensions de représentation usuelles. Dans le cas de très longues séquences, elle est moins efficace que les autres architectures. Cependant, le transformeur toujours de meilleurs résultats pour ces séquences (SHIM & SUNG, 2022).

Type du module	Complexité	Nombre d'Opérations Séquentielles	Longueur du Chemin Emprunté par le Gradient
Auto-Attention	$\mathcal{O}(n^2 \cdot d)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Recurrent	$\mathcal{O}(n \cdot d^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Convolutif	$\mathcal{O}(k \cdot n \cdot d^2)$	$\mathcal{O}(1)$	$\mathcal{O}(\log_k n)$
MLP	$\mathcal{O}(n \cdot d^2)$	$\mathcal{O}(1)$	$+\infty$

TABLE 2.1 – Analyse comparative de la performance des différents types d'architectures S2S.  $n$  est la longueur de la séquence,  $d$  est la dimension de la représentation vectorielle des éléments et  $k$  est la taille du noyau des convolutions (VASWANI et al., 2017, Tab. 1).

## 2.7 Conclusion

Dans ce chapitre, nous avons introduit l'apprentissage S2S, une tâche générale en ML dont la MT et la ASR sont des cas particuliers (SZABO et al., s. d.). Nous avons défini le problème traité en modélisation S2S, expliqué les différentes architectures neuronales utilisées pour le résoudre et donné une comparaison de leurs performances théoriques. Dans le chapitre suivant, nous étudions les deux exemples de MT et ASR dans la mesure où ils peuvent être appliqués à notre problématique.

# Chapitre 3

## Traduction automatique et reconnaissance automatique de la parole

Dans le chapitre 1, nous avons introduit la MT et la ASR comme avenues possibles pour la réhabilitation de la parole chez les patients de l’aphasie de Broca. Ensuite, dans le chapitre 2, nous avons présenté le problème général dont ces deux tâches sont des cas particuliers : celui de la modélisation S2S. Nous y avons posé formellement le problème et présenté les architectures neuronales majeures qui ont été utilisées pour le résoudre en les comparant. Dans ce chapitre, nous abordons dans plus de détails les aspects spécifiques de ces deux tâches. Nous étudions l’application des architectures présentées (notamment le transformeur) dans leur contexte.

### 3.1 Traduction automatique

Étant donné un langage source  $L_S$  sur un vocabulaire  $\Sigma_S$ , un langage cible  $L_C$  sur un vocabulaire  $\Sigma_C$ , et une relation *d’équivalence*<sup>1</sup>  $\sim$  sur  $L_S \cup L_C$  la MT de  $L_S$  en  $L_C$  consiste à trouver une fonction calculable  $f : L_S \rightarrow L_C$  qui vérifie

$$\forall x \in L_S, \quad f(x) \sim x \tag{3.1}$$

la relation  $\sim$  donne un sens d’identité entre les phrases. Sa définition peut varier dans sa rigueur et sa précision, elle est par exemple mathématiquement définie dans le contexte de compilation<sup>2</sup> (HADJ, 2015), elle a une définition floue dans le contexte de la MT du langage naturel (CHAN, 2015) et dans notre contexte de MT pour la correction des erreurs<sup>3</sup> peut être vu comme un cas intermédiaire (BRYANT et al., 2022).

Ce flou dans la définition empêche l’application efficace de la traduction automatique à

- 
1. Dans le sens mathématique du terme, c-à-d. une relation réflexive, symétrique et transitive.
  2. Qui est bien un exemple de MT où  $L_S$  et  $L_C$  sont des langages de programmation et  $\sim$  est la relation d’équivalence sémantique.
  3. Il s’agit encore d’un exemple de MT.

base de règle (RMBT) dans ce contexte. La plupart des succès ont été eus par la traduction automatique neuronale (NMT) (YANG et al., 2020). Ces méthodes s'inscrivent facilement dans le cadre de l'apprentissage S2S tel que nous l'avons défini dans section 2.1. Il est donc naturel de considérer les modèles étudiés dans le chapitre 2 comme des candidats pour la MT. Plus précisément, l'architecture de transformeur est la plus prometteuse en vue de l'analyse comparative effectuée dans le chapitre 2 (voir Table 2.1). De ce constat, nous consacrons cette section à l'étude de l'utilisation des transformateurs pour la NMT.

### 3.1.1 Traduction automatique à base de transformeur

Le transformeur tel que nous l'avons présenté dans section 2.6 peut être utilisé pour la MT. En effet, (VASWANI et al., 2017) l'ont appliqué à cette même tâche. Ayant déjà introduit l'architecture et le principe de fonctionnement du transformeur, nous concentrerons sur les particularités de son application à la MT.

#### Plongement lexical

Le transformeur — comme tout réseau de neurones — n'opère que sur des vecteurs. Pour lui faire passer des phrases, il faut donc les transformer en vecteurs. La première étape consiste à transformer les mots en vecteurs. C'est ce qu'on appelle un *plongement lexical* (ALMEIDA & XEXÉO, 2019).

La première étape dans le calcul des plongements lexicaux est de diviser le texte en unités lexicales (tokens). Il peut s'agir de mots, d'ensembles de mots ou d'unités plus petites qu'un mot. La sortie de cette étape est appelée un *vocabulaire*. Plusieurs techniques existent pour effectuer ce découpage. La plus simple et d'assimiler chaque mot à un token, mais d'autres techniques comme le byte pair encoding (BPE) proposent la segmentation des mots en sous-éléments. Le choix des sous-mots à garder dans le vocabulaire est le plus souvent une fonction de la fréquence dans le corpus (RAI & BORAH, 2021).

Après l'avoir construit, un plongement doit être associé au vocabulaire. L'application qui mappe les tokens sur les vecteurs peut avoir une implémentation arbitraire. Il peut s'agir d'un simple tableau de vecteurs (PASZKE et al., 2019), comme il peut s'agir d'un réseau de neurones (CHURCH, 2017). L'objectif est d'associer aux tokens similaires des vecteurs similaires<sup>4</sup>, on ramène ainsi, la relation mal définie de similarité entre les mots à une relation bien définie sur les vecteurs. Une fois qu'on a mis la phrase sous forme vectorielle, le transformeur peut la traiter (voir Section 2.6).

#### Stratégie de décodage

La sortie de la dernière couche du transformeur est une loi de probabilité sur le vocabulaire de  $L_C$  (voir Figure 3.1). À fin d'obtenir une phrase, le décodeur est appelé autorégressivement avec la phrase vide comme grain. À chaque étape, il génère un mot

---

4. Qui ont des grands produits scalaires.

en utilisant la loi de probabilité. Ce mot est ajouté à la phrase et le décodeur est appelé récursivement avec la phrase ainsi obtenue. Ce processus est répété jusqu'à ce que le symbole de fin de phrase soit généré (VASWANI et al., 2017).

Plusieurs méthodes existent pour générer un mot à partir de la loi de probabilité. La plus simple est de choisir le mot qui a la plus grande probabilité, on parle alors de décodage glouton, décodage argmax ou décodage par maximum a posteriori.

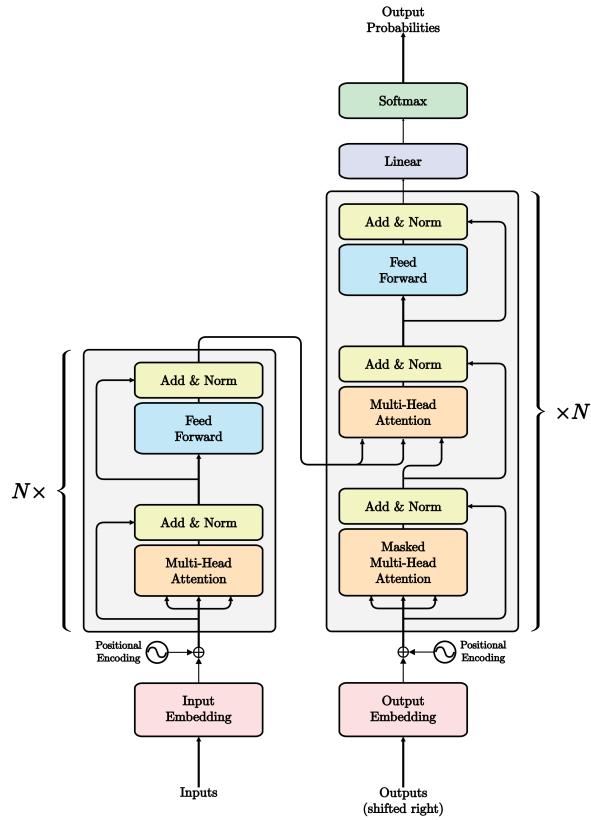


FIGURE 3.1 – Architecture de transformeur (VASWANI et al., 2017).

Cependant, cette méthode est suboptimale. L'alternative la plus courante est ce qu'on appelle le décodage *beam search* (MEISTER et al., 2021).

```

1 def beam_search_decoder(data, k):
2     sequences = [[[], 0.0]]
3     for row in data:
4         all_candidates = []
5         for i in range(len(sequences)):
6             seq, score = sequences[i]
7             for j in range(len(row)):
8                 candidate = [seq + [j], score - log(row[j])]
9                 all_candidates.append(candidate)
10            ordered = sorted(all_candidates, key=lambda tup: tup[1])
11            sequences = ordered[:k]
12    return sequences

```

Algorithme 3.1 – Passe d'un MLP

# Bibliographie

- (s. d.). <https://www.who.int/publications-detail-redirect/WHO-HIS-HSI-Rev.2012.03>
- (s. d.). <https://www.acqol.com.au/instruments>
- ACHARYA, A. B., & WROTON, M. (2022). Broca Aphasia. In *StatPearls*. StatPearls Publishing. <http://www.ncbi.nlm.nih.gov/books/NBK436010/>
- ALMEIDA, F., & XEXÉO, G. (2019). Word Embeddings : A Survey [arXiv :1901.09069 [cs, stat]], (arXiv :1901.09069). <http://arxiv.org/abs/1901.09069>
- ART, S. M. (2013). *English : Functional areas of the brain*. <https://commons.wikimedia.org/wiki/File:Cerveau.jpg>
- BA, J. L., KIROS, J. R., & HINTON, G. E. (2016). Layer Normalization [arXiv :1607.06450 [cs, stat]], (arXiv :1607.06450). <http://arxiv.org/abs/1607.06450>
- BAHDANAU, D., CHO, K., & BENGIO, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate [arXiv :1409.0473 [cs, stat]], (arXiv :1409.0473). <https://doi.org/10.48550/arXiv.1409.0473>
- BARBE, P., & LEDOUX, M. (2012). *Probabilité (L3M1)*. EDP Sciences.
- BASODI, S., JI, C., ZHANG, H., & PAN, Y. (2020). Gradient amplification : An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3), 196-207. <https://doi.org/10.26599/BDMA.2020.9020004>
- BENGIO, Y., SIMARD, P., & FRASCONI, P. (1994). Learning long-term dependencies with gradient descent is difficult. 5(2), 157-166. <https://doi.org/10.1109/72.279181>
- BROCA, M. P. (1861). REMARQUES SUR LE SIÉGE DE LA FACULTÉ DU LANGAGE ARTICULÉ, SUIVIES D'UNE OBSERVATION D'APHÉMIE (PERTE DE LA PAROLE), 18.
- BRODMANN, K. (2007). *Brodmann's : Localisation in the Cerebral Cortex*. Springer Science & Business Media.
- BRYANT, C., YUAN, Z., QORIB, M. R., CAO, H., NG, H. T., & BRISCOE, T. (2022). Grammatical Error Correction : A Survey of the State of the Art [arXiv :2211.05166 [cs]], (arXiv :2211.05166). <http://arxiv.org/abs/2211.05166>
- CALIN, O. (2020). *Deep Learning Architectures*. Springer. <https://link.springer.com/book/10.1007/978-3-030-36721-3>
- CHAN, S.-w. (2015). *Routledge Encyclopedia of Translation Technology*. Routledge, Taylor & Francis Group.
- CHAPEY, R. (2008). *Language Intervention Strategies in Aphasia and Related Neurogenic Communication Disorders*. Wolters Kluwer Health/Lippincott Williams & Wilkins.
- CHO, K., van MERRIENBOER, B., BAHDANAU, D., & BENGIO, Y. (2014). On the Properties of Neural Machine Translation : Encoder-Decoder Approaches [arXiv :1409.1259 [cs, stat]], (arXiv :1409.1259). <http://arxiv.org/abs/1409.1259>

- CHUNG, J., GULCEHRE, C., CHO, K., & BENGIO, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [arXiv :1412.3555 [cs]], (arXiv :1412.3555). <http://arxiv.org/abs/1412.3555>
- CHURCH, K. W. (2017). Word2Vec. *Natural Language Engineering*, 23(1), 155-162. <https://doi.org/10.1017/S1351324916000334>
- CNSA, C. n. d. s. p. l. (2015). session de sensibilisation. [https://www.cnsa.fr/documentation/dossierpressefno\\_021015\\_bd.pdf](https://www.cnsa.fr/documentation/dossierpressefno_021015_bd.pdf)
- COSTANZA, A., AMERIO, A., AGUGLIA, A., MAGNANI, L., SERAFINI, G., AMORE, M., MERLI, R., AMBROSETTI, J., BONDOLFI, G., MARZANO, L., & BERARDELLI, I. (2021). "Hard to Say, Hard to Understand, Hard to Live" : Possible Associations between Neurologic Language Impairments and Suicide Risk. *Brain Sciences*, 11(12), 1594. <https://doi.org/10.3390/brainsci11121594>
- CS480/680 Lecture 19 : Attention and Transformer Networks.* (2019). [https://www.youtube.com/watch?v=OyFJWRnt\\_AY](https://www.youtube.com/watch?v=OyFJWRnt_AY)
- da FONTOURA, D. R., RODRIGUES, J. d. C., CARNEIRO, L. B. d. S., MONÇÃO, A. M., & de SALLLES, J. F. (2012). Rehabilitation of language in expressive aphasias : a literature review. *Dementia & Neuropsychologia*, 6(4), 223-235. <https://doi.org/10.1590/S1980-57642012DN06040006>
- FATHI, S. (2021). *Recurrent Neural Networks*. <https://link.springer.com/book/10.1007/978-3-030-89929-5>
- FLOWERS, H., SKORETZ, S., SILVER, F., ROCHON, E., FANG, J., FLAMAND-ROZE, C., & MARTINO, R. (2016). Poststroke Aphasia Frequency, Recovery, and Outcomes : A Systematic Review and Meta-Analysis. *Archives of Physical Medicine and Rehabilitation*, 97, 2188-2201. <https://doi.org/10.1016/j.apmr.2016.03.006>
- FODOR, J. A. (1983). *The Modularity of Mind* [Google-Books-ID : 0vg0AwAAQBAJ]. MIT Press.
- FUKUSHIMA, K. (1980). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 193-202. <https://doi.org/10.1007/BF00344251>
- HADJ, A. A. e. (2015). *Analyse syntaxique et traduction : outils et techniques cours et exercices résolus*. Ellipses.
- HALLOWELL, B. (2017). *Aphasia and Other Acquired Neurogenic Language Disorders : A Guide for Clinical Excellence*. Plural Publishing.
- HE, K., ZHANG, X., REN, S., & SUN, J. (2016). Deep Residual Learning for Image Recognition, 770-778. [https://openaccess.thecvf.com/content\\_cvpr\\_2016/html/He\\_Deep\\_Residual\\_Learning\\_CVPR\\_2016\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html)
- HOCHREITER, S., & SCHMIDHUBER, J. (1997). Long short-term memory. 9(8), 1735-1780.
- INFORMATIK, F., BENGIO, Y., FRASCONI, P., & SCHMIDHUBER, J. (2003). Gradient Flow in Recurrent Nets : the Difficulty of Learning Long-Term Dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*.
- JACOBS, M., & ELLIS, C. (2021). Estimating the cost and value of functional changes in communication ability following telepractice treatment for aphasia. *PLOS ONE*, 16(9), e0257462. <https://doi.org/10.1371/journal.pone.0257462>
- JDIFOOL, t. p., Mysid. (2006). *Français : Les principaux lobes du cerveau, vue latérale gauche. Inspiré de la figure 728 de Gray's Anatomy*. [https://commons.wikimedia.org/wiki/File:Brain%5C\\_diagram%5C\\_fr.svg](https://commons.wikimedia.org/wiki/File:Brain%5C_diagram%5C_fr.svg)

- KALCHBRENNER, N., & BLUNSMON, P. (2013). Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700-1709. <https://aclanthology.org/D13-1176>
- KALCHBRENNER, N., ESPEHOLT, L., SIMONYAN, K., OORD, A. v. d., GRAVES, A., & KAVUKCUOGLU, K. (2017). Neural Machine Translation in Linear Time [arXiv :1610.10099 [cs]], (arXiv :1610.10099). <http://arxiv.org/abs/1610.10099>
- KAMEOKA, H., TANAKA, K., KWAŚNY, D., KANEKO, T., & NOBUKATSU, H. (2020). ConvS2S-VC : Fully Convolutional Sequence-to-Sequence Voice Conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 1849-1863. <https://doi.org/10.1109/TASLP.2020.3001456>
- KEARNS, M. J., & VAZIRANI, U. (1994). *An Introduction to Computational Learning Theory* [Google-Books-ID : vCA01wY6iywC]. MIT Press.
- LAROCHELLE, H., & HINTON, G. E. (2010). Learning to combine foveal glimpses with a third-order Boltzmann machine. *Advances in Neural Information Processing Systems*, 23. <https://proceedings.neurips.cc/paper/2010/hash/677e09724f0e2df9b6c000b75b5da10d-Abstract.html>
- LAROUSSE. (s. d.). <https://www.larousse.fr/dictionnaires/francais/aphasie/4448>
- LECUN, Y., BOSEN, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., & JACKEL, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4), 541-551. <https://doi.org/10.1162/neco.1989.1.4.541>
- LECUN, Y., BOTTOU, L., BENGIO, Y., & HAFFNER, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. <https://doi.org/10.1109/5.726791>
- LECUN, Y., BENGIO, Y., & HINTON, G. (2015). Deep learning. *Nature*, 521(7553), 436-444. <https://doi.org/10.1038/nature14539>
- LI, X., ZHANG, G., HUANG, H. H., WANG, Z., & ZHENG, W. (2016). Performance Analysis of GPU-Based Convolutional Neural Networks. *2016 45th International Conference on Parallel Processing (ICPP)*, 67-76. <https://doi.org/10.1109/ICPP.2016.15>
- LIU, Z., HUANG, J., XU, Y., WU, J., TAO, J., & CHEN, L. (2021). Cost-effectiveness of speech and language therapy plus scalp acupuncture versus speech and language therapy alone for community-based patients with Broca's aphasia after stroke : a post hoc analysis of data from a randomised controlled trial. *BMJ Open*, 11(9), e046609. <https://doi.org/10.1136/bmjopen-2020-046609>
- LORCH, M. (2011). Re-examining Paul Broca's initial presentation of M. Leborgne : Understanding the impetus for brain and language research. *Cortex*, 47(10), 1228-1235. <https://doi.org/10.1016/j.cortex.2011.06.022>
- LUONG, M.-T., PHAM, H., & MANNING, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation [arXiv :1508.04025 [cs]], (arXiv :1508.04025). <http://arxiv.org/abs/1508.04025>
- MARTINS, A. (2018). Lecture 9 : Machine Translation and Sequence-to-Sequence Models.
- MEISTER, C., VIEIRA, T., & COTTERELL, R. (2021). If beam search is the answer, what was the question ? [arXiv :2010.02650 [cs]], (arXiv :2010.02650). <http://arxiv.org/abs/2010.02650>
- MOHAMMED, N., NARAYAN, V., PATRA, D. P., & NANDA, A. (2018). Louis Victor Leborgne ("Tan"). *World Neurosurgery*, 114, 121-125. <https://doi.org/10.1016/j.wneu.2018.02.021>

- MORRISON, M. (2016). I would tell you if I could : Language loss, depression, and the challenge of treating patients with aphasia. *8*(1).
- MUKHERJEE, A. (2021). A Study of the Mathematics of Deep Learning [arXiv :2104.14033 [cs, math, stat]], (arXiv :2104.14033). <http://arxiv.org/abs/2104.14033>
- National Aphasia Association.* (s. d.). <https://www.aphasia.org/>
- Noyaux gris centraux. (s. d.). <https://www.msdmanuals.com/fr/professional/multimedia/figure/noyaux-gris-centraux>
- OPPENHEIM, A. V., & SCHAFER, R. W. (2013). *Discrete-time Signal Processing*. Pearson.
- PALLAVI, J., PERUMAL, R. C., & KRUPA, M. (2018). Quality of Communication Life in Individuals with Broca's Aphasia and Normal Individuals : A Comparative Study. *Annals of Indian Academy of Neurology*, *21*(4), 285-289. [https://doi.org/10.4103/aian.AIAN\\_489\\_17](https://doi.org/10.4103/aian.AIAN_489_17)
- PASCANU, R., MIKOLOV, T., & BENGIO, Y. (s. d.). On the difficulty of training recurrent neural networks.
- PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., ... GARNETT, R. (2019). PyTorch : An Imperative Style, High-Performance Deep Learning Library. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- RAI, A., & BORAH, S. (2021). Study of Various Methods for Tokenization. In J. K. MANDAL, S. MUKHOPADHYAY & A. ROY (Éd.), *Applications of Internet of Things* (p. 193-200). Springer. [https://doi.org/10.1007/978-981-15-6198-6\\_18](https://doi.org/10.1007/978-981-15-6198-6_18)
- ROSS, K., & WERTZ, R. (2010). Quality of life with and without aphasia. *Aphasiology*. <https://doi.org/10.1080/02687030244000716>
- SCIENCES, N. A. o., MEDICINE, I. o., & ACKERMAN, S. (1992). *Discovering the Brain* [Google-Books-ID : 3ypUq9nuncQC]. National Academies Press.
- SEBASTIAN, R., & MIRJALILI, V. (2017). *Python Machine Learning : Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow* (2<sup>e</sup> éd., T. 1). Packt Publishing.
- SHIM, K., & SUNG, W. (2022). A Comparison of Transformer, Convolutional, and Recurrent Neural Networks on Phoneme Recognition [arXiv :2210.00367 [cs, eess]], (arXiv :2210.00367). <http://arxiv.org/abs/2210.00367>
- SMAÏLI, K., LANGLOIS, D., & PRIBIL, P. (2022). Language rehabilitation of people with BROCA aphasia using deep neural machine translation. *Fifth International Conference Computational Linguistics in Bulgaria*, 162.
- STAHLBERG, F. (2020). Neural Machine Translation : A Review. *Journal of Artificial Intelligence Research*, *69*, 343-418. <https://doi.org/10.1613/jair.1.12007>
- SZABO, V., PLESIAK, M., YANG, Y., & HEUMANN, C. (s. d.). *Modern Approaches in Natural Language Processing*. [https://sls-lmu.github.io/seminar\\_nlp\\_ss20/index.html](https://sls-lmu.github.io/seminar_nlp_ss20/index.html)
- TAN, X., CHEN, J., LIU, H., CONG, J., ZHANG, C., LIU, Y., WANG, X., LENG, Y., YI, Y., HE, L., SOONG, F., QIN, T., ZHAO, S., & LIU, T.-Y. (2022). NaturalSpeech : End-to-End Text to Speech Synthesis with Human-Level Quality [arXiv :2205.04421 [cs, eess]], (arXiv :2205.04421). <http://arxiv.org/abs/2205.04421>
- VASWANI, A., SHAZER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, & POLOSUKHIN, I. (2017). Attention is All you Need. *Advances in Neural*

*Information Processing Systems, 30.* [https://proceedings.neurips.cc/paper/2017/  
hash/3f5ee243547dee91fb053c1c4a845aa-Abstract.html](https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fb053c1c4a845aa-Abstract.html)

YANG, S., WANG, Y., & CHU, X. (2020). A survey of deep learning techniques for neural machine translation. *arXiv preprint arXiv :2002.07526*.