



المدرسة الوطنية للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État 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

Encadré par :

Pr. SMAILI Kamel
smaili@loria.fr
Dr. LANGLOIS David
david.langlois@loria.fr
Dr. ZAKARIA Chahnez
c_zakaria@esi.dz

Table des matières

Page de garde	i
Table des matières	iii
Table des figures	v
Liste des algorithmes	vi
Sigles et abréviations	vii
Introduction générale	1
1 Notions générales	3
1.1 Aphasicie de Broca	3
1.1.1 Histoire et anatomie	3
1.1.2 Classification des syndromes aphasiq	6
1.1.3 Causes, prévalence et incidence	7
1.1.4 Effet sur la qualité de vie	7
1.1.5 Traitement	8
1.2 Traduction automatique	8
1.2.1 Généralités	9
1.2.2 Classification	9

1.3	Reconnaissance automatique de la parole	10
1.4	Conclusion	11
2	Apprentissage séquence-à-séquence	13
2.1	Énoncé du problème	13
2.2	Perceptorns multicouches	14
2.2.1	Généralités	14
2.2.2	Application à la modélisation de séquence	15
2.2.3	Avantages et inconvénients	15
2.3	Architecture encodeur-décodeur	16
2.4	Réseaux de neurones récurrents	17
2.4.1	Réseaux de neurones récurrents simples	18
2.4.2	Portes, gated recurrent unit et long short-term memory	20
2.5	Réseau de neurones à convolutions	22
2.6	Transformeurs	25
2.6.1	Architecture générale	25
2.6.2	Encodage positionnel	26
2.6.3	Couches attention	27
2.6.4	Autres éléments de l'architecture	29
2.6.5	Analyse comparative de la performance	29
2.7	Conclusion	30
3	Traduction automatique et reconnaissance automatique de la parole	31
3.1	Traduction automatique	31
3.1.1	Traduction automatique à base de transformeur	32
3.1.2	Generative pre-trained transformer (GPT)	35

3.1.3	Bidirectional encoder representations from transformers (BERT)	37
3.1.4	Bidirectional auto-regressive transformer (BART)	38
3.2	Reconnaissance automatique de la parole	39
3.2.1	Wav2Vec	39
3.2.2	Whisper	39
Bibliographie		46

Table des figures

1.1	Cerveau de Victor Louis Leborgne avec la lésion encadrée.	4
1.2	Encéphale humain.	5
1.3	Division morphologique et fonctionnelle du cerveau.	5
1.4	Classification des syndromes aphasiques classiques.	6
1.5	Comparaison de la qualité de vie de communication chez les individus saints et ceux qui souffrent de l'aphasie de Broca.	8
1.6	Taxonomie des méthodes de traduction automatique.	9
1.7	Triangle de Vauquois.	10
1.8	Taxonomie des techniques d'ASR.	11
2.1	Architecture sous-jacente d'un MLP de profondeur 4.	14
2.2	Architecture encodeur-décodeur	17
2.3	RNN v.s FFN	17
2.4	Dépliement temporel d'un RNN sur une entrée de longueur 4.	19
2.5	Dépliement temporel d'un encodeur-décodeur récurrent.	19
2.6	Forme générale d'un RNN à portes.	20
2.7	Architecture interne d'un GRU	21
2.8	Architecture interne d'un LSTM	22
2.9	Couche convolutive unidimensionnelle.	24
2.10	Recurrent Continuous Translation Model	24

2.11	Architecture de ByteNet.	24
2.12	Architecture de ConvS2S.	24
2.13	L'architecture de transformeur.	26
2.14	Matrice d'encodage de la position pour $r = 10^4$ et $d = 512$	27
2.15	Schéma d'une couche attention multitête.	28
3.1	L'architecture de transformeur.	35
3.2	Exemple de décodage par beam search.	36
3.3	Architecture de GPT.	37
3.4	Architecture de BERT.	37
3.5	Architecture de BART	38
3.6	Affinement de BART pour la traduction.	39
3.7	Architecture de Wav2Vec.	40
3.8	Architecture de Whisper.	40

Liste des algorithmes

2.1	Passe d'un MLP	15
2.2	Passe d'un RNN	18
2.3	Encodeur-décodeur récurrent.	20
3.1	Byte pair encoding.	33
3.2	Score BLEU.	34
3.3	Décodage par beam search.	36

Sigles et abréviations

ASR	reconnaissance automatique de la parole
AVC	accident vasculaire cérébrale
BART	bidirectional auto-regressive transformer
BERT	bidirectional encoder representations from transformers
BLEU	bilingual evaluation understudy
BPE	byte pair encoding
BPTT	rétro-propagation dans le temps
CLM	modélisation causale du langage
CNN	réseau de neurones à convolutions
DL	apprentissage profond
FFN	réseau de neurones feed-forward
GPT	generative pre-trained transformer
GRU	gated recurrent unit
IL	interlingue
LC	langage cible
LLM	grand modèle de langage
LS	langue source
LSTM	long short-term memory
ML	apprentissage automatique
MLM	modélisation masquée du langage
MLP	perceptron multicouches
MT	traduction automatique
NLP	traitement automatique du langage
NMT	traduction automatique neuronale
NSP	prédiction de la prochaine phrase
RMBT	traduction automatique à base de règle

RNN	réseau de neurones récurrent
S2S	séquence-à-séquence
SMT	traduction automatique statistique

Introduction générale

L'aphasie est un trouble linguistique qui complique un grand pourcentage d'accidents vasculaires cérébraux, une condition médicale qui touche plus de 12.2 millions de personnes par an. Ce chiffre est susceptible d'augmenter avec l'augmentation de l'espérance de vie (FEIGIN et al., 2022).

L'aphasie de Broca est une forme d'aphasie qui affecte la capacité de s'exprimer oralement ou par écrit. Elle résulte d'une lésion dans l'aire de Broca, une région du cerveau qui est responsable de la production de la parole. Les personnes qui ont l'aphasie de Broca ont des difficultés à produire des mots, mais peuvent comprendre ce qui est dit (CHAPEY, 2008).

Ces difficultés peuvent avoir des conséquences néfastes sur plusieurs aspects de la vie quotidienne. Ceci peut inclure la communication avec les proches, la participation à des activités sociales, l'exercice d'un emploi ou même la demande d'aide en cas d'urgence (HALLOWELL, 2017).

L'utilisation de techniques d'apprentissage automatique et du traitement automatique du langage au bénéfice des personnes atteintes de l'aphasie est une piste de recherche qui a capturé l'attention de plusieurs chercheurs (MISRA et al., 2022 ; QIN et al., 2022 ; SMAÏLI et al., 2022).

Dans ce mémoire, notre objectif est de fournir une revue de l'état de l'art sur les travaux qui ont été faits dans cette direction. Nous portons une attention particulière à la traduction automatique et à la reconnaissance automatique de la parole appliquées à l'aphasie de Broca.

Pour ce faire, nous avons organisé notre travail en trois chapitres :

1. Notions générales.

Dans ce chapitre, nous présentons en général les domaines de recherche qui nous intéressent. À cette fin, le chapitre est divisé en trois sections :

- (1) La première section présente l'aphasie de Broca,
- (2) la deuxième introduit la traduction automatique
- (3) et la troisième présente la reconnaissance automatique de la parole.

2. Apprentissage séquence-à-séquence.

Ce chapitre sert à familiariser le lecteur avec le cadre d'étude général dans lequel s'inscrivent la traduction automatique, la reconnaissance automatique de la parole

et la majorité des tâches de traitement automatique du langage. Il s'agit de la modélisation de séquences. Nous y présentons l'énoncé du problème et les différentes architectures neuronales qui ont été proposées pour le résoudre.

3. Traduction automatique et reconnaissance automatique de la parole.

Ce chapitre part de l'étude générale faite dans le chapitre 2. Il détaille l'application des meilleures architectures neuronales qui y sont présentées dans le cadre de la traduction automatique et de la reconnaissance automatique de la parole.

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 deux avenues de recherche qui nous semblent pertinentes pour résoudre ce problème. Il s'agit respectivement de la traduction automatique (MT, de l'anglais : machine translation) et de la reconnaissance automatique de la parole (ASR, de l'anglais : automatic speech recognition). Dans les deux chapitres suivants, nous développons en plus de détails ces deux approches en explorant la littérature scientifique là-dessus.

1.1 Aphasie de Broca

L'aphasie ; emprunté au Grec ancien $\alpha\varphiασία$ 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

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 communiquera 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 (α 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).

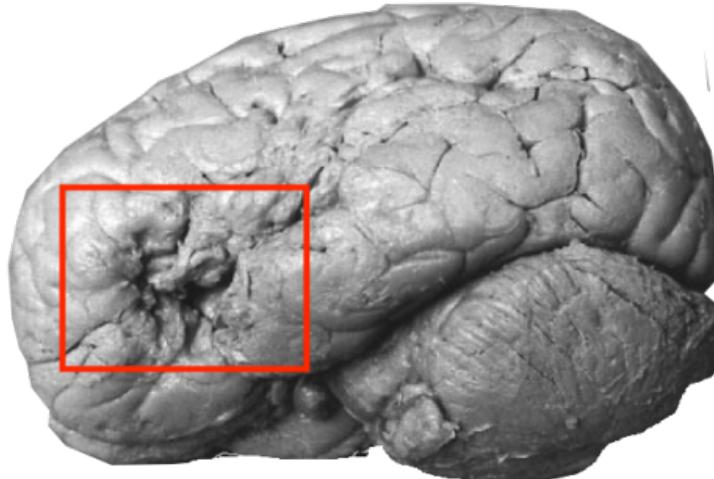


FIGURE 1.1 – Cerveau de Victor Louis Leborgne avec la lésion encadrée (LOPES, 2019).

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).

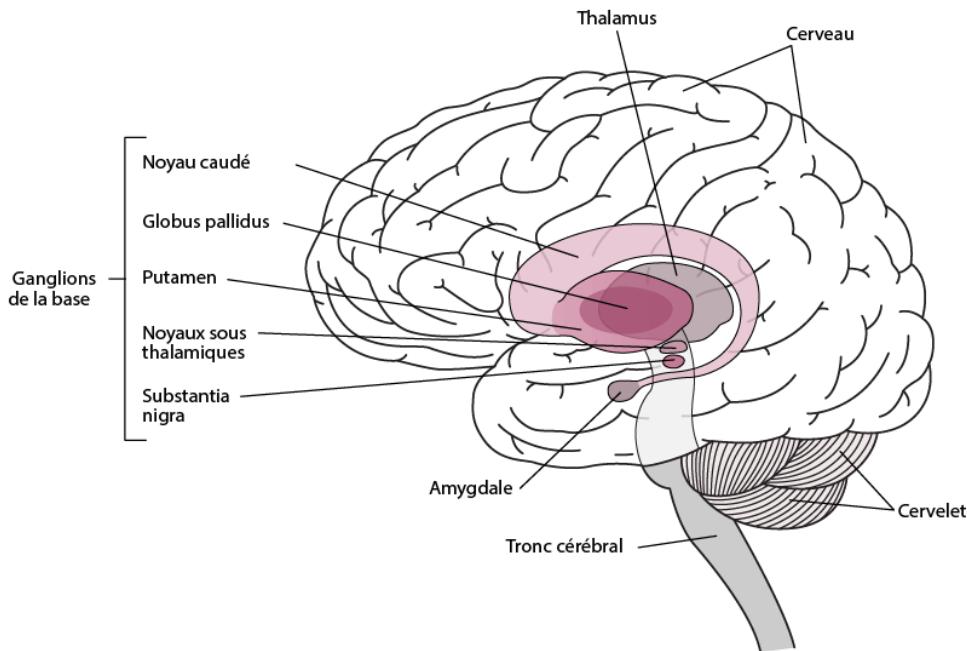
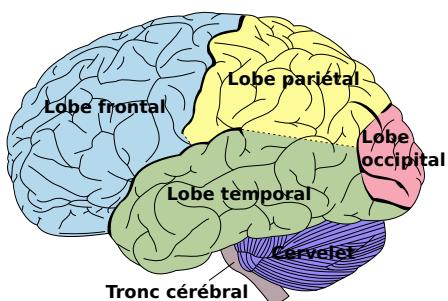
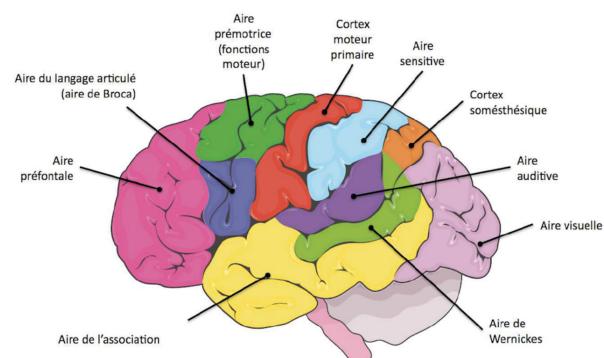


FIGURE 1.2 – Encéphale humain (« Noyaux gris centraux », s. d.).

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 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 (ou *syndromes*) dissimilaires en cause (région touchée du cerveau) et effet (conséquences pour la communication) (HALLOWELL, 2017). La table 1.1 et la figure 1.4 présentent une classification des syndromes aphasiques classiques. En particulier, notre sujet d'intérêt,

Syndrome Aphasique	Expressive / Réceptive	Localisation de la Lésion	Effet sur la Compréhension	Effet sur l'Expression
Aphasie de Wernicke	Réceptive	Aire de Wernicke (Brodmann 22 ¹)	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	Réceptive	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)

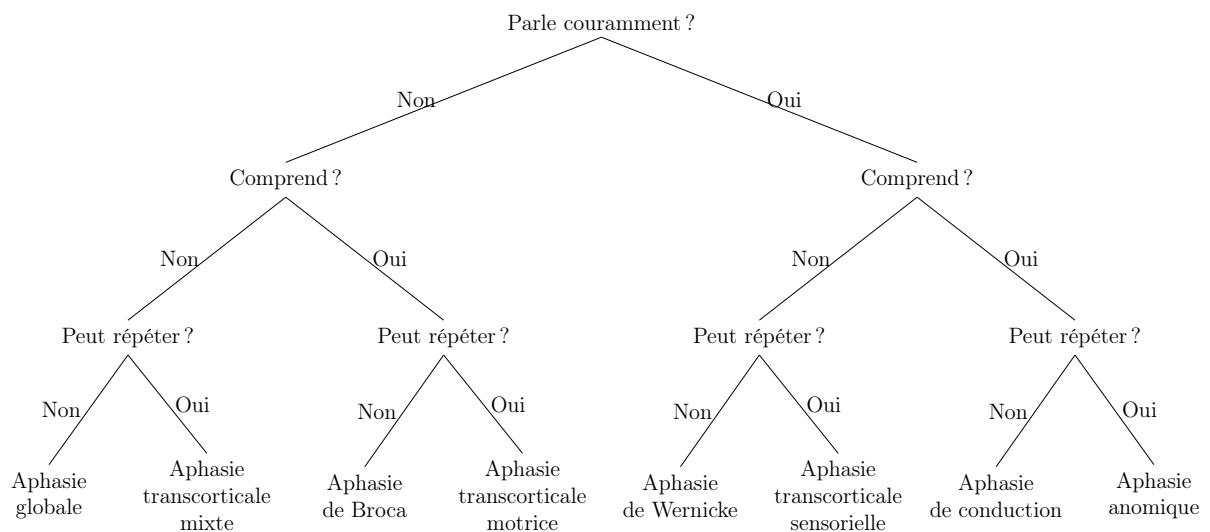


FIGURE 1.4 – Classification des syndromes aphasiques classiques (SREEDHARAN, 2018).

l'aphasie 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. BRODMANN, 2007.

1.1.3 Causes, prévalence et incidence

Les AVC sont la première cause d'aphasie (HALLOWELL, 2017). En effet, 30% des individus atteints d'un AVC développent une aphasic (FLOWERS et al., 2016). Parmi ces individus, 43% ont une aphasic 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'aphasic par an (une incidence d'un peu moins de 0.05%). Inversement, 75% des cas d'aphasic sont causés par un AVC (CNSA, 2015).

Il est difficile d'estimer l'incidence et la prévalence globales de l'aphasic. 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'aphasic (« National Aphasia Association », s. d.), 2 millions Américains en souffrent, une prévalence de 0.6%. En France, ce chiffre est de l'ordre de 300000 cas, 30000 desquels sont nouveaux (CNSA, 2015). Ceci donne une prévalence de 0.44% et un taux d'incidence 0.044%.

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

1.1.4 Effet sur la qualité de vie

Il n'est pas surprenant que l'aphasic ait un impact négatif sur la qualité de vie des individus qui en souffrent. En effet, les individus atteints de l'aphasic ont une moins bonne qualité de vie que les individus en bonne santé selon les critères WHOQOL-BREF² et WPI³(voir Table 1.2).

Measure	Mean	Range	SD	Mean difference	95% C. I. ^a 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 aphasic (ROSS & WERTZ, 2010).

En particulier, l'aphasic de Broca diminue mesurablement la qualité de vie des individus qu'elle affecte dans toute tâche qui nécessite la communication (PALLAVI et al., 2018). La Table 1.5 le montre bien pour les trois exemples d'activités sociales, confiances en soi et capacité à réaliser ses responsabilités quotidiennes.

2. <https://www.who.int/publications-detail-redirect/WHO-HIS-HSI-Rev.2012.03>

3. <https://www.acqol.com.au/instruments>

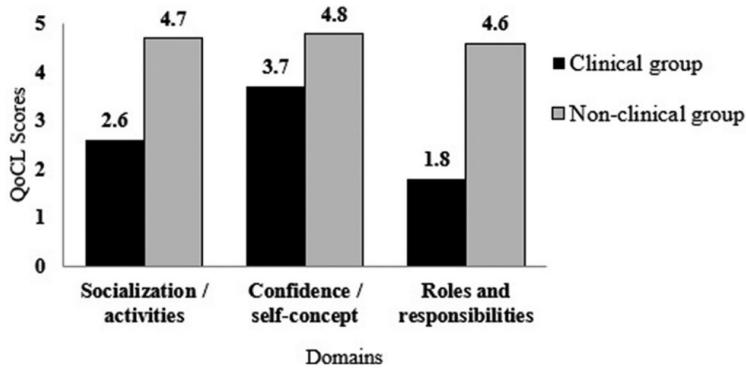


FIGURE 1.5 – Comparaison de la qualité de vie de communication chez les individus saints et ceux qui souffrent de l'aphasie de Broca (PALLAVI et al., 2018).

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.

1.1.5 Traitement

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 ; Z. 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 Traduction automatique

Nous avons établi que les difficultés d'accès aux traitements pour l'aphasie de Broca constituent un risque inadmissible en vue de ses effets potentiellement catastrophiques. Il est donc urgent de trouver des solutions pour y remédier. Une solution automatique à base de MT semble être une piste intéressante à explorer.

1.2.1 Généralités

La MT est une branche du traitement automatique du langage (NLP, de l'anglais : natural language processing). Elle étudie l'utilisation des systèmes informatiques pour traduire le texte ou la parole d'une langue (appelée source) vers une autre (appelée cible) (CHAN, 2015). Dans cette section, nous introduisons la MT du texte pour donner un point de référence aux discussions des chapitres suivants.

1.2.2 Classification

Plusieurs méthodes de MT sont présentées dans la littérature. Ces méthodes peuvent être classées selon les outils mathématiques qu'elles utilisent. On distingue ainsi trois grandes familles de méthodes de MT (YANG et al., 2020) :

1. Des méthodes basées sur des connaissances linguistiques (règles de traduction).
2. Des méthodes basées sur les statistiques.
3. Des méthodes basées sur les réseaux de neurones.

On les appelle respectivement traduction automatique à base de règle (RMBT, de l'anglais : rule-based machine translation), traduction automatique statistique (SMT, de l'anglais : statistical machine translation), traduction automatique neuronale (NMT, de l'anglais : neural machine translation).

À l'intérieur de ces familles, les méthodes peuvent être distinguées en sous-familles. Ceci donne lieu à la hiérarchie représentée par la Figure 1.6

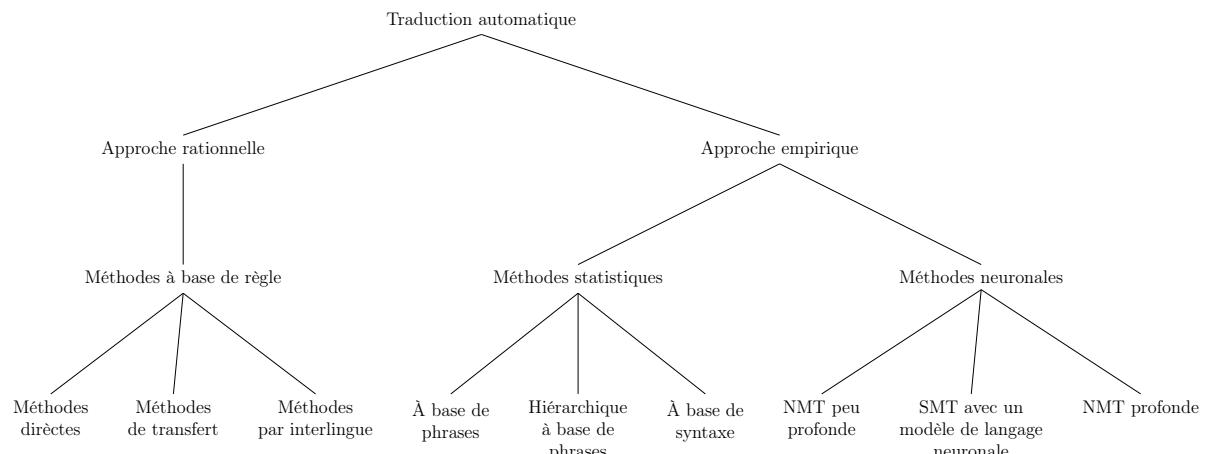


FIGURE 1.6 – Taxonomie des méthodes de traduction automatique (COSTA-JUSSÀ et al., 2016 ; YANG et al., 2020).

Une autre classification récurrente dans la littérature est celle du triangle de Vauquois (voir Figure 1.7). Elle se base sur l'utilisation ou non de représentations intermédiaires des langues. Si la phrase traduite est construite directement à partir de la phrase à traduire, il s'agit de traduction *directe*. Ce paradigme présente deux majeurs inconvénients. Le premier — et le plus évident — est la difficulté de passer directement d'une langue à une

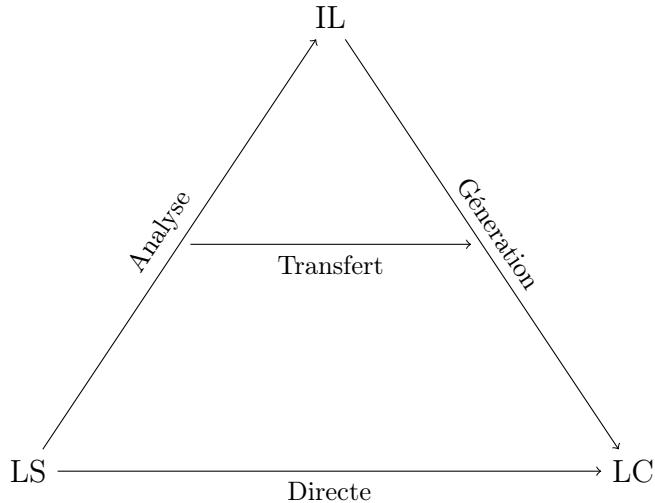


FIGURE 1.7 – Triangle de Vauquois (COSTA-JUSSÀ et al., 2016).

autre. Le deuxième est la scalabilité : pour maintenir la traduction entre n langues, $\frac{n(n-1)}{2}$ couples de fonctions de traduction sont nécessaires (CHAN, 2015).

Les méthodes indirectes se distinguent en méthodes “par transfert”, qui utilisent une représentation intermédiaire dépendant de la langue (par exemple, arbre de syntaxe) et en méthodes “par interlingue”, qui utilisent la même représentation intermédiaire pour toutes les langues. Ce dernier paradigme résout le problème de scalabilité (seulement n couples de traductions sont nécessaires) au coût de construire une interlingue assez riche à représenter toute phrase dans toute langue (CHAN, 2015).

Dans ce travail, nous nous intéressons principalement aux sous arbre droit de la Figure 1.6. En effet, la majorité de notre investigation porte sur son dernier élément (la NMT profonde). Ceci est motivé par l’énorme succès dont le apprentissage profond (DL, de l’anglais : deep learning) a fait preuve dans la dernière décennie (SEBASTIAN & MIRJALILI, 2017).

1.3 Reconnaissance automatique de la parole

L’ASR est une autre branche du NLP. Son but est d’extraire automatiquement d’un signal audio qui représente une parole la représentation textuelle de cette parole (HUANG et al., 2021).

Les méthodes d’ASR peuvent être divisées en méthodes basées sur l’acoustique, méthodes basées sur la reconnaissance de motifs et méthodes basées sur l’intelligence artificielle (voir Figure 1.8).

Comme pour la MT, notre intérêt porte principalement sur les méthodes à base d’intelligence artificielle. Plus spécifiquement sur les méthodes qui utilisent le DL.

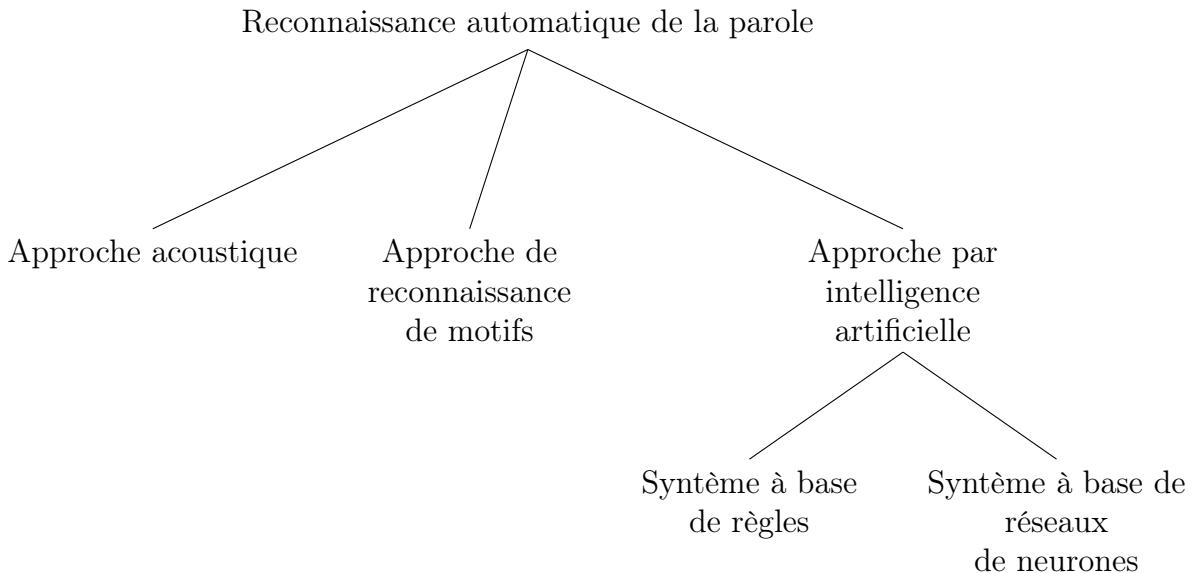


FIGURE 1.8 – Taxonomie des techniques d’ASR (VOLNY et al., 2012).

1.4 Conclusion

Dans ce chapitre, nous avons fait le tour des sujets abordés dans ce mémoire. Nous avons introduit le problème central, ces spécificités, contexte historique et scientifique et des chemins possibles vers une solution.

Nous avons commencé par introduire le lecteur aux troubles linguistiques qui sont l’aphasie en général et l’aphasie de Broca en particulier. Nous l’avons abordé sous différents angles, notamment celui de l’histoire et de la neuroanatomie. Nous l’avons également mis dans le contexte des autres syndromes aphasiques. Une discussion des causes de l’aphasie de Broca a également été menée. Des statistiques sur l’incidence et la prévalence de l’aphasie de Broca, ainsi que des études sur ses conséquences ont été présentées pour aider le lecteur à apprécier l’ampleur du problème et sa gravité.

Nous avons terminé notre introduction à l’aphasie de Broca par une discussion des traitements typiquement déployés contre elle. 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.

Après cela, nous avons introduit les techniques de NLP que nous jugeons pertinentes pour la résolution de notre problème : la MT et la ASR. Pour la MT, nous avons commencé par une description de la tâche. En suite, nous avons avancé les classifications de ses méthodes qu’on trouve dans la littérature. Notre choix d’exploration s’est porté sur la classe que nous avons trouvé la plus prometteuse : la NMT.

Pour la ASR, nous avons repris la même démarche. Après avoir décrit la tâche, nous avons illustré une classification des méthodes. Nous avons encore une fois choisi de nous concentrer sur les techniques à base de DL.

Dans le reste de ce document, nous étudions en plus de détail la NMT et la ASR. Le

chapitre suivant aborde l'apprentissage séquence-à-séquence (S2S), le problème général dont ces deux tâches sont des sous-problèmes. Celui qui suit rentre dans le détail de l'application de la NMT et de la ASR.

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 φ_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^{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 $(w_0, w_1, w_2, w_3, w_4) = (4, 5, 7, 5, 4)$.

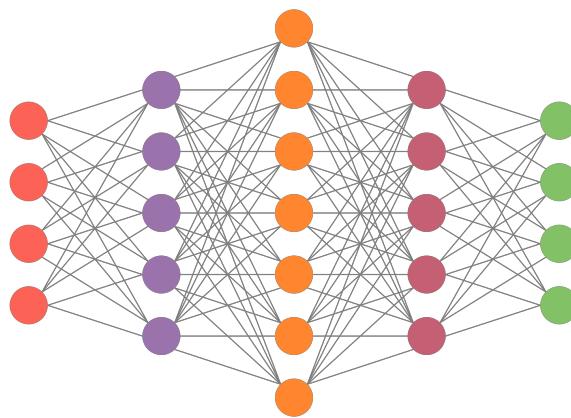


FIGURE 2.1 – Architecture sous-jacente d'un MLP de profondeur 4.

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 régler¹ 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).

Algorithme 2.1 Passe d'un MLP

```

1 def mlp_s2s(xs: Sequence, k: int) -> Sequence:
2     ys = [] # Initialiser la sortie
3     for x in blocks(xs, k): # Parcourir les blocs
4         y = mlp(x) # Traiter chaque bloc
5         ys.append(y) # Concatener les resultats
6     return ys

```

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 com-

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

prendre 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.² 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.

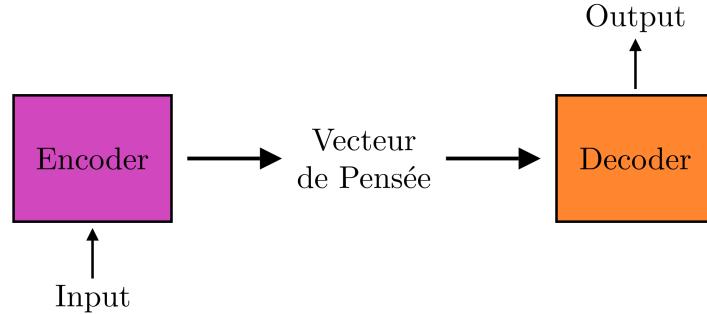


FIGURE 2.2 – Architecture encodeur-décodeur

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

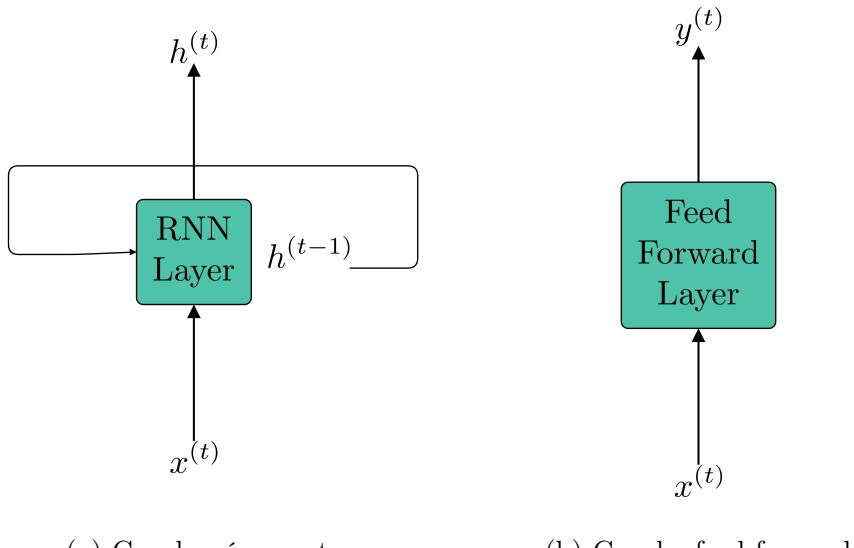


FIGURE 2.3 – RNN v.s FFN

comportement, l'état d'un RNN est décalé d'une unité est réinjecté dans l'entrée (voir Figure 2.3a et Algorithme 2.2). Cela est très différent des MLP qui n'ont pas de boucle de rétroaction, il s'agit de réseaux de neurones feed-forward (FFN, de l'anglais : feed-forward

Algorithme 2.2 Passe d'un RNN

```
1 def rnn_pass(xs: Sequence, h=0) -> Vector:
2     for x in xs: # pour chaque element de la sequence
3         # traitement de l'element + mise a jour de l'état
4         h = process(x, h)
5     return h # retourne l'état final
```

network) (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 temps³, $x^{(t)}$, $h^{(t)}$ sont respectivement l'entrée et la sortie à l'instant t , n est la longueur de la séquence et φ est la fonction d'activation (FATHI, 2021). Dans le cas où φ 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 impulsionale 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).

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).

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 et Algorithme 2.3).

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

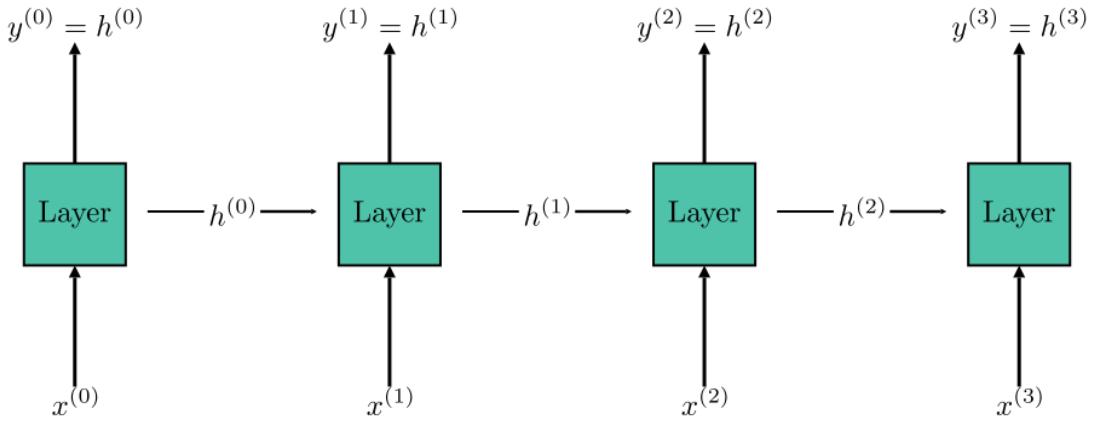


FIGURE 2.4 – Dépliement temporel d'un RNN sur une entrée de longueur 4.

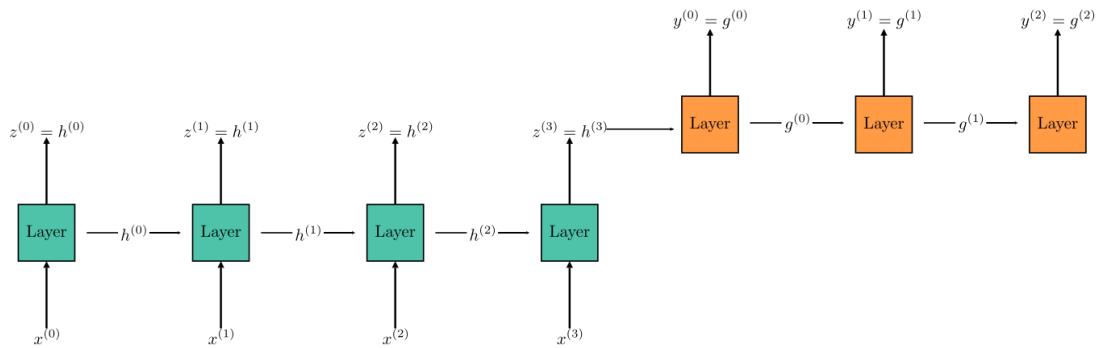


FIGURE 2.5 – Dépliement temporel d'un encodeur-décodeur récurrent sur une entrée de longueur 4 et une sortie de longueur 3.

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'appelle 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).

Algorithme 2.3 Encodeur–décodeur récurrent.

```

1 def rnn_s2s(xs:Sequence) -> Sequence:
2     encoder = RNN() # encodeur recurrent
3     decoder = RNN() # decodeur recurrent
4     ys = [] # sequence de sortie initialement vide
5     h = encoder(xs) # encode la sequence d'entree
6
7     while True:
8         # decoder sequentiellement
9         h = decoder(0, h)
10        ys.append(h)
11        if h == eos: # si on a genere le symbole de fin
12            # on arrete la generation
13            break
14    return ys

```

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).

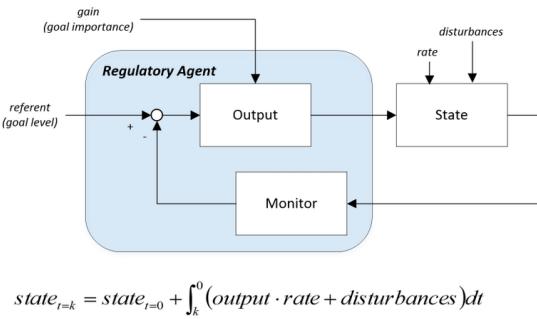


FIGURE 2.6 – Forme générale d'un RNN à portes.

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).

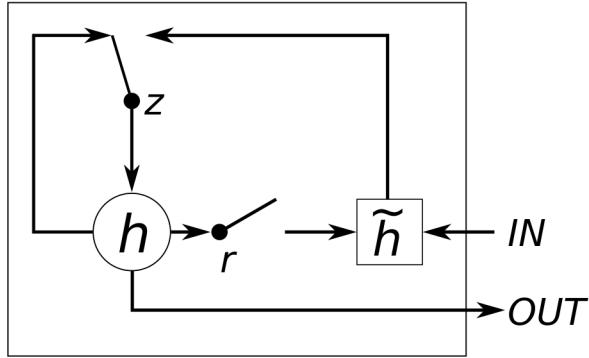


FIGURE 2.7 – Architecture interne d'un GRU (CHUNG et al., 2014, Fig. 1b)

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 σ est la fonction *sigmoïde*⁴ et \odot et le produit d'Hadamard⁵.

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

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

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

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

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 φ 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'oublier (f) et une porte de sortie (o) (voir Figure 2.8).

Le fonctionnement des portes est similaire à celui du GRU. Les équations 2.14–2.19 le montrent en détails⁶ (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. $\tanh : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$

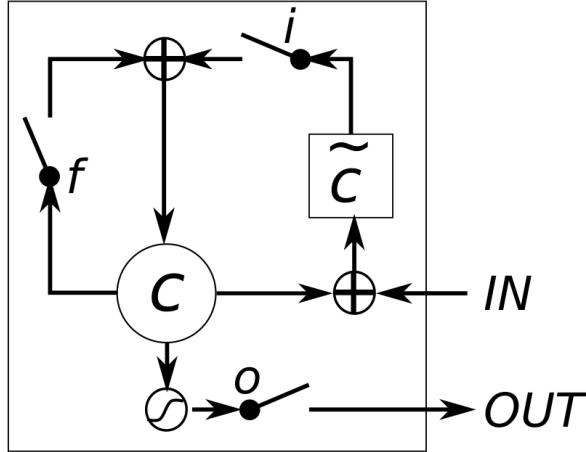


FIGURE 2.8 – Architecture interne d'un LSTM (CHUNG et al., 2014, Fig. 1a)

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

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

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

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

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

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

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 :⁷

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

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 par fois de représentation *hiérarchique*.

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).

7. $g^- : t \mapsto g(-t)$.

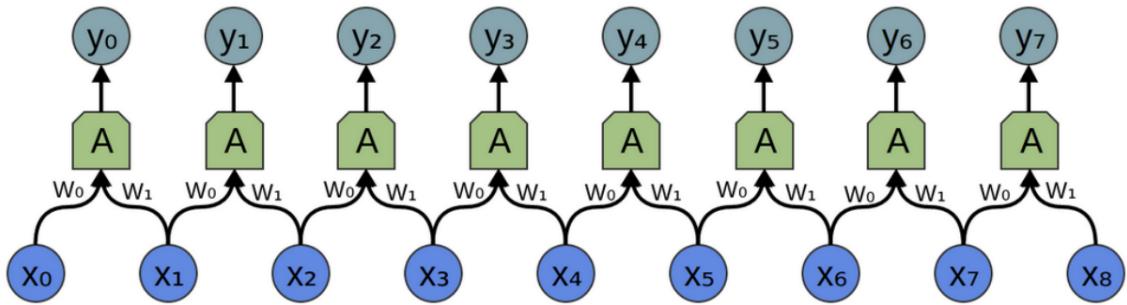


FIGURE 2.9 – Couche convulsive unidimensionnelle.

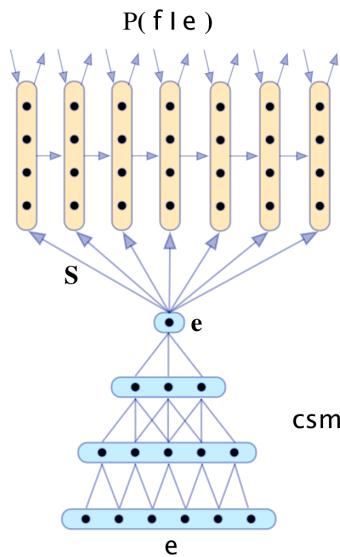


FIGURE 2.10 – Recurrent Continuous Translation Model

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.

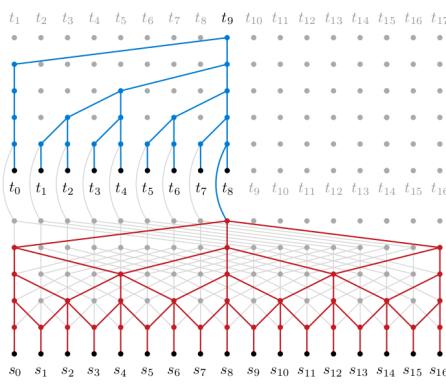


FIGURE 2.11 – Architecture de ByteNet.

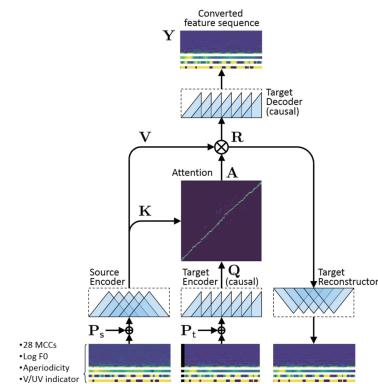


FIGURE 2.12 – Architecture de ConvS2S.

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 (X. LI et al., 2016).

Or, la nature locale du produit de convolution rend difficile aux CNN d'apprendre les corrélations globales⁸. 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.

2.6 Transformateurs

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 couche⁹. 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.

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

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

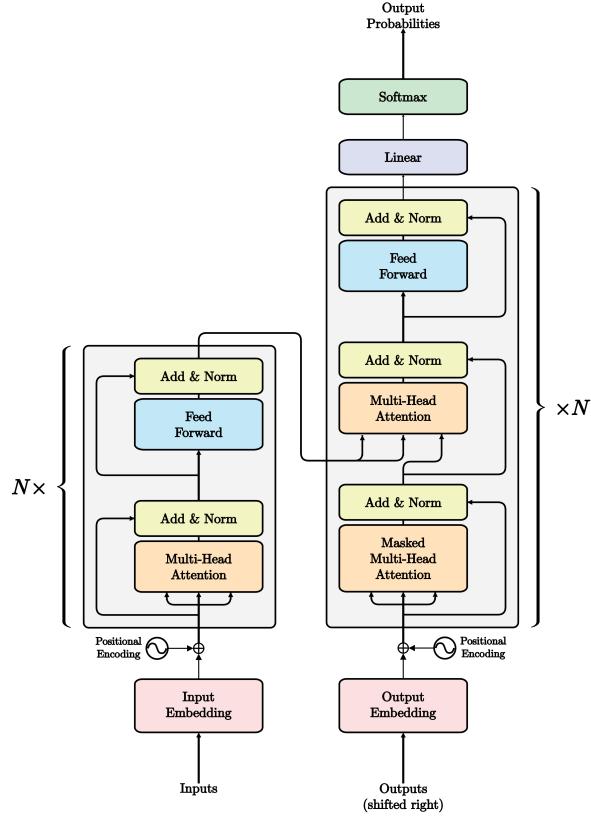


FIGURE 2.13 – Architecture de transformeur (VASWANI et al., 2017, Fig 1).

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 permutations¹⁰. La sortie de la couche de plongement 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 :

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

10. 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 encodages identiques.

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 $d = 512$.

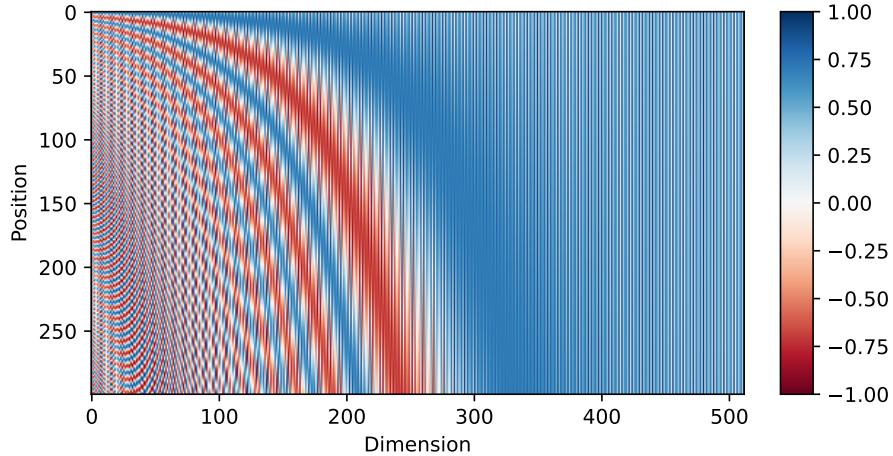


FIGURE 2.14 – Matrice d'encodage de la position pour $r = 10^4$ et $d = 512$.

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 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 ¹¹ de dimensions respectives $d_Q \times l, d_K \times m, d_V \times n$, qui représentent chacune l'encodage d'une séquence¹². 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

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

12. 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.

la requête Q et la clé K . Il représente la similarité entre les deux. La fonction softmax¹³ le normalise pour obtenir des poids positifs de somme 1. Enfin, le résultat et 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'apprentissage¹⁴ (VASWANI et al., 2017).

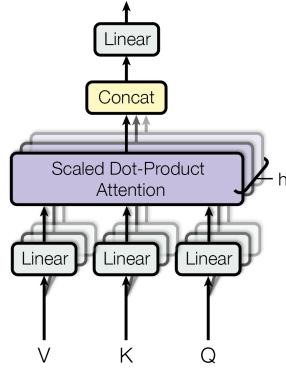


FIGURE 2.15 – Schéma d'une couche attention multi-tête (VASWANI et al., 2017, Fig 2).

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 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

13. softmax : $\mathbb{R}^p \rightarrow \mathbb{R}^p$, $x = (x_1, x_2, \dots, x_p) \mapsto \frac{e^x}{\sum_{i=1}^p e^{x_i}}$

14. 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).

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 Σ_S , un langage cible L_C sur un vocabulaire Σ_C , et une relation *d’équivalence*¹ \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 compilation² (HADJ, 2015), elle a une définition floue dans le contexte de la MT du langage naturel (CHAN, 2015) et notre contexte de MT pour la correction des erreurs³, peut être vu comme un cas intermédiaire (BRYANT et al., 2022).

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 là encore d’un exemple de MT. L_S est une copie de L_C avec un vocabulaire augmenté. En fonction de la complexité des erreurs, la relation \sim peut ressembler à l’égalité.

Ce flou dans la définition empêche l'application efficace de la RMBT dans ce contexte. La plupart des succès ont été eus par la 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, de suites 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. Cette technique présente l'inconvénient de produire des vocabulaires très grands. Il est possible de limiter la taille du vocabulaire en ne gardant que les mots les plus fréquents, remplaçant les autres par un token spécial [UNK]], mais cela a pour conséquence de perdre l'information sur les mots rares. Dans un contexte correction d'erreurs, où les erreurs intrinsèquement rares, cela est inacceptable (RAI & BORAH, 2021).

D'autres techniques tentent de réduire la taille du vocabulaire en utilisant des tokens plus petits qu'un mot. On parle de techniques de tokenisation en sous-mots. L'une des plus connues est le byte pair encoding (BPE) (SENNRICH et al., 2016). Pour construire le vocabulaire, BPE commence par découper les mots en caractères (les bytes initiaux). Ensuite, il regroupe le couple de bytes les plus fréquents en un seul byte, remplaçant toutes leurs occurrences par ce dernier. Cette opération est répétée jusqu'à ce que le nombre de bytes soit égal au nombre de tokens désiré ou pour un nombre maximal d'itérations (voir Algorithme 3.1).

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

Algorithme 3.1 Byte pair encoding (SENNRICH et al., 2016).

```
1 def get_stats(vocab):
2     pairs = defaultdict(int)
3     for word, freq in vocab.items():
4         symbols = word.split()
5         for i in range(len(symbols)-1):
6             pairs[symbols[i], symbols[i+1]] += freq
7     return pairs
8
9
10 def merge_vocab(pair, v_in):
11     v_out = []
12     bigram = re.escape(" ".join(pair))
13     p = re.compile(r"(?<!\S)" + bigram + r"(?!\\S)")
14     for word in v_in:
15         w_out = p.sub("".join(pair), word)
16         v_out[w_out] = v_in[word]
17     return v_out
18
19
20 vocab = {
21     "l o w </w>": 5,
22     "l o w e r </w>": 2,
23     "n e w e s t </w>": 6,
24     "w i d e s t </w>": 3
25 }
26 num_merges = 10
27
28 for _ in range(num_merges):
29     pairs = get_stats(vocab)
30     best = max(pairs, key=pairs.get)
31     vocab = merge_vocab(best, vocab)
32     print(best)
```

vecteurs similaires⁴, 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).

Évaluation et métriques

Plusieurs métriques peuvent être utilisées pour évaluer la qualité d'une traduction automatique. Les plus simples sont les métriques qui sortent de la matrice de confusion (notamment la précision et le rappel). Ces métriques sont conçues pour quantifier la qualité des classifications simples. Cela les rend inadaptées à la MT. Par exemple, la précision de la traduction candidate “Le le le le” par rapport à la référence “Le chat est sur le tapi” est égale à 1.

Des métriques plus adaptées à la MT ont été proposées. La plus connue parmi elles est bilingual evaluation understudy (BLEU) (PAPINENI et al., 2002). BLEU prend la précision

4. Qui ont des grands produits scalaires.

comme point de départ. Il l'améliore en apportant deux modifications : 1. il considère la précision sur les n -grammes⁵ plutôt que sur les mots individuels et 2. il prend en compte le nombre de fois où un n -gramme est présent dans la référence. Le résultat est appelé *la précision n -gramme modifiée*. Il est donné par la formule suivante⁶ :

$$p_n(\hat{y}, y) = \frac{\sum_{g \in G_n(\hat{y})} \min(\#(\hat{y}, g), \#(y, g))}{\sum_{g \in G_n(\hat{y})} \#(\hat{y}, g)} \quad (3.2)$$

BLEU est obtenu en prenant la moyenne géométrique des p_n multipliée par une pénalité exponentielle pour les phrases trop courtes (voir Algorithme 3.2).

Algorithme 3.2 Score BLEU.

```

1 def ngrams(sequence, n):
2     output = []
3     for i in range(len(sequence) - n + 1):
4         output.append(sequence[i:i + n])
5     return output
6
7
8 def modified_precision(candidate, truth, n):
9     candidate_ngrams = ngrams(candidate, n)
10    truth_ngrams = ngrams(truth, n)
11    count = 0
12    for ngram in candidate_ngrams:
13        if ngram in truth_ngrams:
14            count += 1
15            truth_ngrams.remove(ngram)
16    return count / len(candidate_ngrams)
17
18
19 def bleu(candidate, truth):
20     pns = []
21     for n in range(len(candidate)):
22         pns.append(modified_precision(candidate, truth, n + 1))
23     log_pns = [log(p_n) for p_n in pns]
24     brevity_penalty = 1
25     if len(candidate) > len(truth):
26         brevity_penalty = exp(1 - len(candidate) / len(truth))
27     return brevity_penalty * exp(sum(log_pns) / len(candidate))

```

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 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).

5. séquences de n mots consécutifs.

6. $G_n(s)$ est l'ensemble des n -grammes de s et $\#(s, s')$ est le nombre d'occurrences de s' dans s .

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.

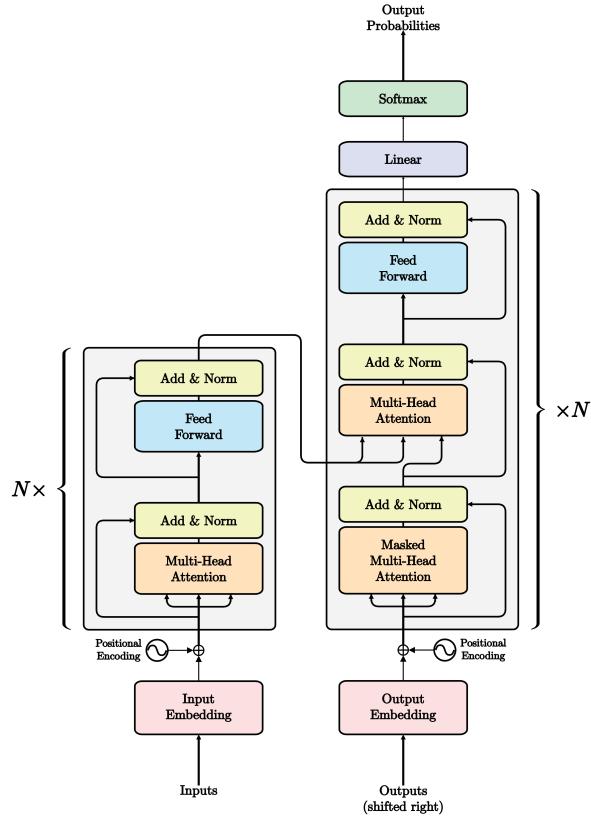


FIGURE 3.1 – Architecture de transformeur (VASWANI et al., 2017).

Cependant, cette méthode est suboptimale. L’alternative la plus courante est ce qu’on appelle le décodage par *beam search* (MEISTER et al., 2021). Le principe est de garder les b meilleurs candidats à chaque étape. À l’étape suivante, b nouveaux candidats sont générés à partir de chacun des b candidats gardés (ce qui donne b^2 candidats). Les b meilleurs candidats sont gardés et le processus est répété jusqu’à ce que le symbole de fin de phrase soit généré. (voir Algorithme 3.3 et Figure 3.2).

3.1.2 Generative pre-trained transformer (GPT)

Generative pre-trained transformer (GPT) est un grand modèle de langage (LLM, de l’anglais : large language model) de OpenAI (RADFORD et al., 2018). Un modèle de langage est une loi de probabilité \mathbb{P} sur les phrases d’une langue (CHAN, 2015). Dans le cas de GPT, cette loi est calculée par un réseau de neurones. Plus précisément, il s’agit d’une *pile de décodeur de transformeur*⁷ (voir Figure 3.3).

7. Une composition de N modules de décodeur. Dans (RADFORD et al., 2018), $N = 12$ (117M paramètres).

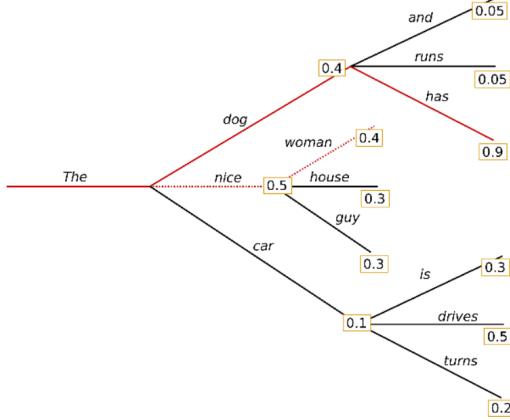


FIGURE 3.2 – Exemple de décodage par beam search (MEISTER et al., 2021).

Algorithme 3.3 Décodage par 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

```

L’entraînement de GPT passe par deux étapes. Dans un premier temps, il est pré-entraîné d’une façon autosupervisée sur un corpus de texte brut. Puis, il est affiné sur un corpus de texte annoté. Cette deuxième étape est un exemple d’apprentissage supervisé classique. La première est beaucoup plus intéressante. Pendant cette étape, le modèle est entraîné sur la tâche de modélisation causale du langage (CLM, de l’anglais : causal language modeling) i.e, la tâche de prédire à partir d’une suite de tokens, le token suivant (RADFORD et al., 2018). Plus exactement, le modèle est entraîné pour maximiser la fonction :

$$L_1 = \sum_{i=k+1}^{n-k} \log \mathbb{P}(t_i | t_{i-k}, \dots, t_{i-1}; \theta) \quad (3.3)$$

où (t_1, t_2, \dots, t_n) est un corpus de taille n , k est un hyperparamètre qui détermine la taille de la fenêtre de contexte et θ représente les paramètres entraînables du modèle.

Le succès de GPT a motivé la création de plusieurs variantes. Notamment, GPT–2 (RADFORD et al., 2019) et GPT–3 (BROWN et al., 2020). La principale différence entre ces variantes est la taille du modèle (1.5B et 175B paramètres respectivement). Un autre modèle basé sur GPT (plus précisément sur GPT–2) qui est très pertinent pour notre travail est GPT–D. Il s’agit d’une copie de GPT–2 dont les paramètres ont été délibérément dégradés à fin de modéliser les erreurs linguistiques associées à la démence (C. LI et al.,

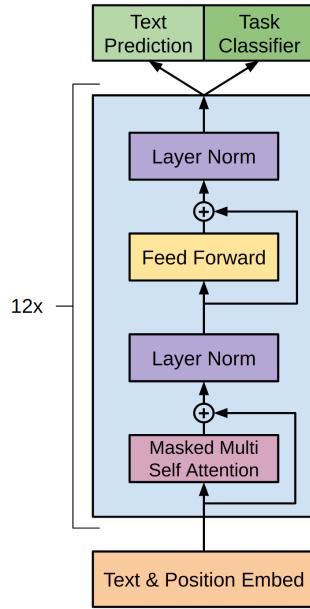


FIGURE 3.3 – Architecture de GPT (RADFORD et al., 2018).

2022).

3.1.3 Bidirectional encoder representations from transformers (BERT)

Bidirectional encoder representations from transformers (BERT) est un LLM pré-entraîné de Google. Il s'agit d'une *pile d'encodeur de transformeur* (voir Figure 3.4). BERT existe en deux versions : *base*, constitué de 12 couches d'encodeur et *large*, constitué de 24. Ils ont respectivement 110M et 340M de paramètres (DEVLIN et al., 2019).

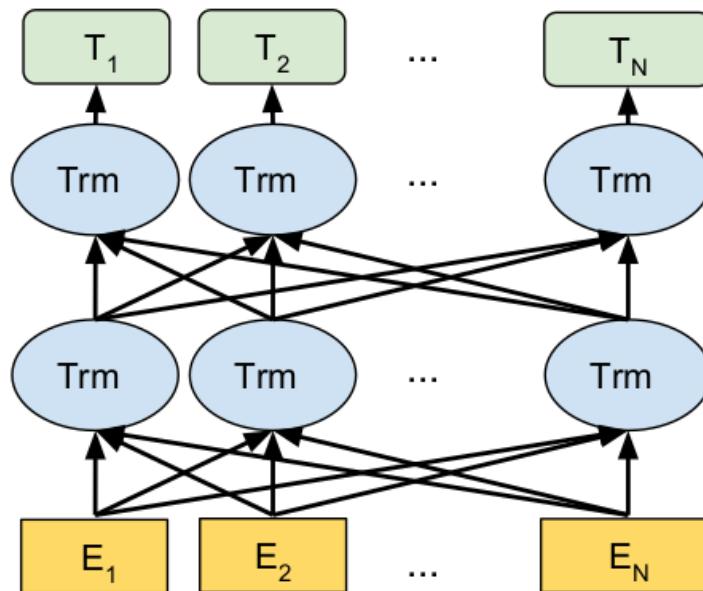


FIGURE 3.4 – Architecture de BERT (DEVLIN et al., 2019)

Tout comme GPT, BERT est pré-entraîné d'une manière autosupervisée puis affiné sur une tâche spécifique. Pour le pré-entraînement de BERT, deux tâches sont utilisées. La première est la modélisation masquée du langage (MLM, de l'anglais : masked language modeling). Dans cette tâche, un pourcentage de tokens est aléatoirement remplacé par un token spécial [MASK]. Le modèle doit alors prédire les tokens cachés en prenant le reste de la phrase comme contexte. La deuxième tâche est la prédiction de la prochaine phrase (NSP, de l'anglais : next sentence prediction). Pour cette tâche, deux phrases sont concaténées et le modèle doit prédire si la deuxième phrase est la suite de la première (DEVLIN et al., 2019).

À son tour, BERT a motivé l'apparition de plusieurs variantes. En effet, il est l'un des modèles les plus réimplémentés dans la littérature à raison d'être open-source. Parmi les variantes, on peut citer : RoBERTa (Y. LIU et al., 2019), qui est identique à BERT sauf qu'il utilise un processus de pré-entraînement différent, CamemBERT (MARTIN et al., 2020), qui est une version française de BERT.

Plusieurs travaux dans la littérature ont utilisé BERT pour la NMT (CLINCHANT et al., 2019 ; ZHU et al., 2020). Certains l'ont même utilisé pour la représentation des phrases aphasiques pour la classification (QIN et al., 2022).

3.1.4 Bidirectional auto-regressive transformer (BART)

Bidirectional auto-regressive transformer (BART) est un LLM de Facebook AI. Contrairement à GPT et BERT, BART est un transformeur S2S. Il utilise BERT et GPT comme encodeur et décodeur respectivement (voir Figure 3.5).

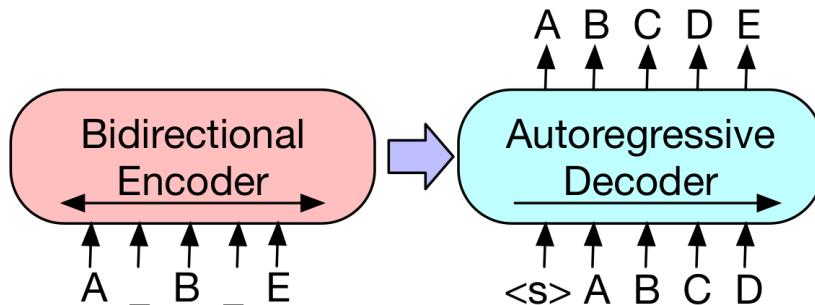


FIGURE 3.5 – Architecture de BART (LEWIS et al., 2019).

À l'instar de GPT et BERT, l'entraînement de BART est constitué d'une phase de pré-entraînement autosupervisé, suivie d'une phase d'affinement. BART est pré-entraîné sur la tâche d'auto-encodage du texte bruité. Similairement à la MLM, un extrait textuel est modifié avant d'être passé à BART. Le modèle est alors chargé de reconstruire le texte original. Cette tâche est plus difficile que la MLM, car le but est de reconstruire la séquence plutôt que de prédire les tokens masqués, mais aussi, car l'ensemble des modifications possibles est plus grand que l'application d'un masque. Il inclut également : la suppression d'un token (sans le remplacer par [MASK]), le remplacement d'une suite de token par [MASK] et l'application d'une permutation de phrases (LEWIS et al., 2019).

L'affinement dépend de la tâche cible. BART peut être affiné pour les tâches de classifi-

cation (comme BERT), génération de texte (comme GPT) ou traduction (voir Figure 3.6).

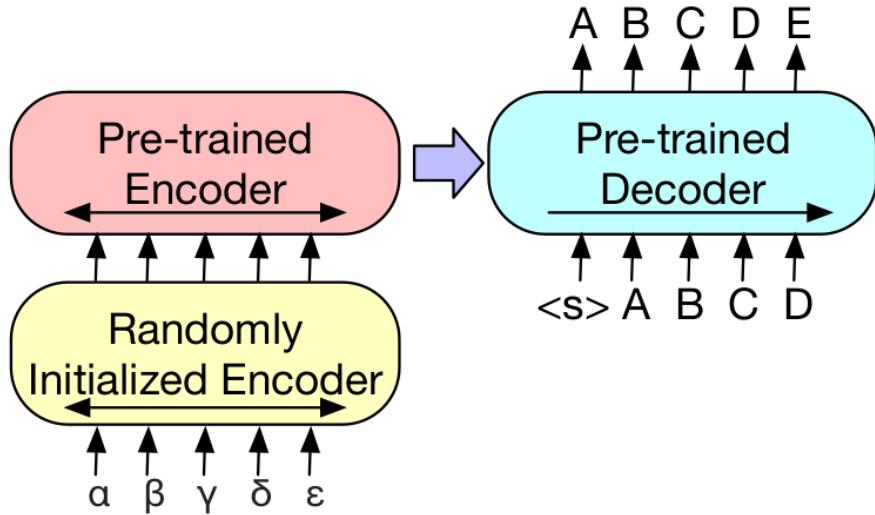


FIGURE 3.6 – Affinement de BART pour la traduction (LEWIS et al., 2019).

3.2 Reconnaissance automatique de la parole

L’ASR est un scénario d’apprentissage S2S où l’ensemble de départ est celui des signaux numériques audio $s \in \mathbb{R}^*$ ⁸, et l’ensemble d’arrivée est un langage L sur un vocabulaire Σ . Étant une tâche d’apprentissage S2S, l’ASR se fie très naturellement au traitement par transformateurs. En effet, il est plus facile de les appliquer à cette tâche, car l’entrée est déjà un vecteur. Plusieurs travaux ont investigué l’application des transformateurs à l’ASR. Deux travaux en particulier ont eu un succès retentissant. Il s’agit de (SCHNEIDER et al., 2019) et de (RADFORD et al., 2022).

3.2.1 Wav2Vec

Wav2Vec (SCHNEIDER et al., 2019) est un modèle de ASR proposé par Facebook AI. Son architecture est composée d’un CNN qui extrait des caractéristiques de l’entrée audio, suivi d’un transformateur qui effectue le traitement séquentiel (voir Figure 3.7).

3.2.2 Whisper

Whisper (RADFORD et al., 2022) est un modèle de ASR proposé par OpenAI. À l’instar de Wav2Vec, son architecture comporte un CNN et un transformateur. Cependant, contrairement à Wav2Vec, l’entrée n’est pas la forme temporelle du signal audio. Il s’agit

8. On rappelle que $A^* := \bigcup_{n \in \mathbb{N}} A^n$.

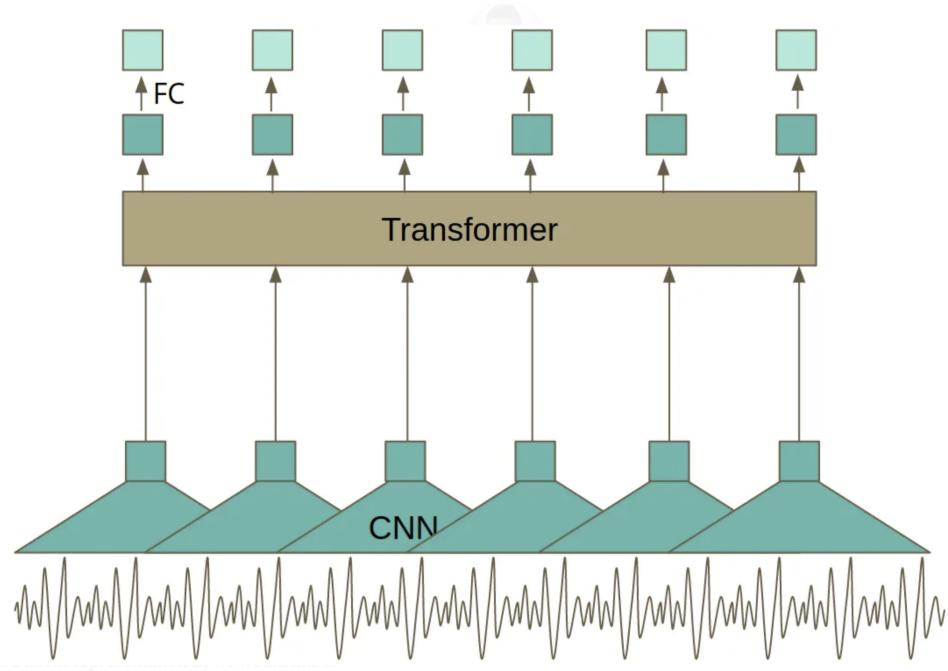


FIGURE 3.7 – Architecture de Wav2Vec (SUS, 2021).

plutôt de représentation spectrale (voir Figure 3.8). Whisper utilise un encodage positionnel sinusoïdal pour l'audio (similaire à (VASWANI et al., 2017)) et un encodage positionnel appris pour le texte (voir section 2.6).

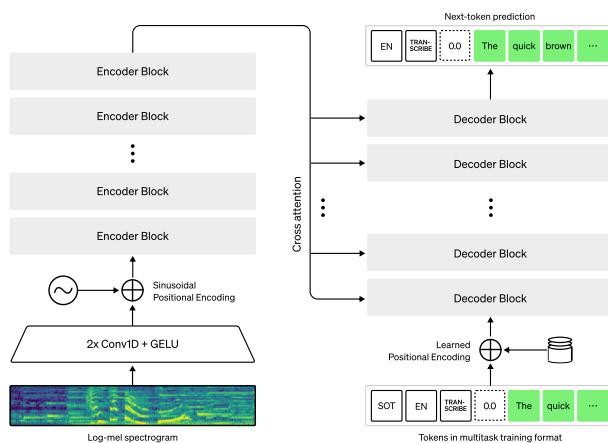


FIGURE 3.8 – Architecture de Whisper (RADFORD et al., 2022).

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.
- BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D., WU, J., WINTER, C., . . . AMODEI, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901. <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>
- 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>
- CLINCHANT, S., JUNG, K. W., & NIKOULINA, V. (2019). On the use of BERT for Neural Machine Translation [arXiv :1909.12744 [cs]], (arXiv :1909.12744). <http://arxiv.org/abs/1909.12744>
- CNSA, C. n. d. s. p. l. (2015). session de sensibilisation. https://www.cnsa.fr/documentation/dossierpressefno_021015_bd.pdf
- COSTA-JUSSÀ, M. R., RAPP, R., LAMBERT, P., EBERLE, K., BANCHS, R. E., & BABYCH, B. (Éd.). (2016). *Hybrid Approaches to Machine Translation*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-21311-8>
- 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
- da FONTOURA, D. R., RODRIGUES, J. d. C., CARNEIRO, L. B. d. S., MONÇÃO, A. M., & de SALLES, 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>
- DEVLIN, J., CHANG, M.-W., LEE, K., & TOUTANOVA, K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding [arXiv :1810.04805 [cs]], (arXiv :1810.04805). <http://arxiv.org/abs/1810.04805>
- FATHI, S. (2021). *Recurrent Neural Networks*. <https://link.springer.com/book/10.1007/978-3-030-89929-5>
- FEIGIN, V. L., BRAININ, M., NORRVING, B., MARTINS, S., SACCO, R. L., HACKE, W., FISHER, M., PANDIAN, J., & LINDSAY, P. (2022). World Stroke Organization (WSO) : Global Stroke Fact Sheet 2022. *International Journal of Stroke : Official Journal of the International Stroke Society*, 17(1), 18-29. <https://doi.org/10.1177/17474930211065917>
- 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
- HOCHREITER, S., & SCHMIDHUBER, J. (1997). Long short-term memory. 9(8), 1735-1780.
- HUANG, W.-C., HAYASHI, T., WU, Y.-C., KAMEOKA, H., & TODA, T. (2021). Pretraining Techniques for Sequence-to-Sequence Voice Conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 745-755. <https://doi.org/10.1109/TASLP.2021.3049336>
- 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
- 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>

- LEWIS, M., LIU, Y., GOYAL, N., GHAZVININEJAD, M., MOHAMED, A., LEVY, O., STOYANOV, V., & ZETTLEMOYER, L. (2019). BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension [arXiv :1910.13461 [cs, stat]], (arXiv :1910.13461). <http://arxiv.org/abs/1910.13461>
- LI, C., KNOPMAN, D., XU, W., COHEN, T., & PAKHOMOV, S. (2022). GPT-D : Inducing Dementia-related Linguistic Anomalies by Deliberate Degradation of Artificial Neural Language Models [arXiv :2203.13397 [cs]], (arXiv :2203.13397). <http://arxiv.org/abs/2203.13397>
- 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, Y., OTT, M., GOYAL, N., DU, J., JOSHI, M., CHEN, D., LEVY, O., LEWIS, M., ZETTLEMOYER, L., & STOYANOV, V. (2019). RoBERTa : A Robustly Optimized BERT Pretraining Approach [arXiv :1907.11692 [cs]], (arXiv :1907.11692). <http://arxiv.org/abs/1907.11692>
- 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>
- LOPES, M. (2019). O paciente “Tan”. <https://www.astropt.org/2019/05/02/o-paciente-tan/>
- 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>
- MARTIN, L., MULLER, B., ORTIZ SUÁREZ, P. J., DUPONT, Y., ROMARY, L., de la CLERGERIE, É., SEDDAH, D., & SAGOT, B. (2020). CamemBERT : a Tasty French Language Model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7203-7219. <https://doi.org/10.18653/v1/2020.acl-main.645>
- 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>
- MISRA, R., MISHRA, S. S., & GANDHI, T. K. (2022). Assistive Completion of Agrammatic Aphasic Sentences : A Transfer Learning Approach using Neurolinguistics-based Synthetic Dataset [arXiv :2211.05557 [cs, q-bio]], (arXiv :2211.05557). <http://arxiv.org/abs/2211.05557>
- 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
- PAPINENI, K., ROUKOS, S., WARD, T., & ZHU, W.-J. (2002). Bleu : a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311-318. <https://doi.org/10.3115/1073083.1073135>
- 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>
- QIN, Y., LEE, T., KONG, A. P. H., & LIN, F. (2022). Aphasia Detection for Cantonese-Speaking and Mandarin-Speaking Patients Using Pre-Trained Language Models. *2022 13th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 359-363. <https://doi.org/10.1109/ISCSLP57327.2022.10037929>
- RADFORD, A., KIM, J. W., XU, T., BROCKMAN, G., MCLEAVEY, C., & SUTSKEVER, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision [arXiv :2212.04356 [cs, eess]], (arXiv :2212.04356). <http://arxiv.org/abs/2212.04356>
- RADFORD, A., NARASIMHAN, K., SALIMANS, T., & SUTSKEVER, I. (2018). Improving Language Understanding by Generative Pre-Training.
- RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., & SUTSKEVER, I. (2019). Language Models are Unsupervised Multitask Learners.
- 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
- ROSS, K., & WERTZ, R. (2010). Quality of life with and without aphasia. *Aphasiology*. <https://doi.org/10.1080/02687030244000716>
- SCHNEIDER, S., BAEVSKI, A., COLLOBERT, R., & AULI, M. (2019). wav2vec : Unsupervised Pre-training for Speech Recognition [arXiv :1904.05862 [cs]], (arXiv :1904.05862). <http://arxiv.org/abs/1904.05862>
- 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^e éd., T. 1). Packt Publishing.
- SENNRICH, R., HADDOW, B., & BIRCH, A. (2016). Neural Machine Translation of Rare Words with Subword Units [arXiv :1508.07909 [cs]], (arXiv :1508.07909). <http://arxiv.org/abs/1508.07909>

- 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.
- SREEDHARAN, S. (2018). *REAL-TIME FMRI BASED NEUROFEEDBACK FOR REHABILITATION OF POST-STROKE PATIENTS WITH APHASIA* (thèse de doct.). <https://doi.org/10.13140/RG.2.2.10868.37760/1>
- STAHLBERG, F. (2020). Neural Machine Translation : A Review. *Journal of Artificial Intelligence Research*, 69, 343-418. <https://doi.org/10.1613/jair.1.12007>
- SUS, (2021). Wav2Vec 2.0 : A Framework for Self-Supervised Learning of Speech Representations. <https://towardsdatascience.com/wav2vec-2-0-a-framework-for-self-supervised-learning-of-speech-representations-7d3728688cae>
- 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
- VASWANI, A., SHAZEER, 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/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- VOLNY, P., NOVAK, D., & ZEZULA, P. (2012). Employing Subsequence Matching in Audio Data Processing.
- YANG, S., WANG, Y., & CHU, X. (2020). A survey of deep learning techniques for neural machine translation. *arXiv preprint arXiv :2002.07526*.
- ZHU, J., XIA, Y., WU, L., HE, D., QIN, T., ZHOU, W., LI, H., & LIU, T.-Y. (2020). Incorporating BERT into Neural Machine Translation [arXiv :2002.06823 [cs]], (arXiv :2002.06823). <http://arxiv.org/abs/2002.06823>