

---

*Rapport de stage de fin d'études :  
Génération de texte*

---



Maxime Toquebiau  
ING5

Février - Août 2020

## Rapport de stage de fin d'études

à retourner au plus tard **1 semaine avant la date de la soutenance**

<b>Rapport de Stage</b> <b>Ing5 - 3<sup>ème</sup> année Cycle Ingénieur</b> <b>Promotion 2020</b>	
<b>Nom : Toquebiau Prénom :</b>  <b>Maxime</b>	<b>Majeure :</b> <ul style="list-style-type: none"><li>• Energie</li><li>• CYB</li><li>• BDA</li><li>• SE</li><li>• VAC</li><li>• Santé</li><li>• Finance</li><li>• OCRS</li></ul>
<b>Entreprise d'accueil</b>  Nom : Ministère des Armées  Adresse : 60 boulevard du général Martial Valin  Adresse du lieu de stage si différent : -  Engagement de Confidentialité : <input type="checkbox"/> oui ✓ non  Rapport Confidentiel à remettre au tuteur de stage à l'issue de la soutenance : <input type="checkbox"/> oui ✓ non  Pénalités Observées :	
<b>Description de la mission</b> <ul style="list-style-type: none"><li>• Réaliser un état de l'art des techniques de génération automatique de texte</li><li>• Étudier plus spécifiquement la génération de conversations</li><li>• Réaliser un prototype de système conversationnel à partir d'une ou plusieurs approches pertinentes</li></ul>	

**EVALUATION DE L'ENTREPRISE**  
*à compléter par le tuteur de stage*  
à insérer dans le rapport de stage **Obligatoirement**

**Fiche d'évaluation du stage**  
**3<sup>ème</sup> année Cycle Ingénieur**  
**Promotion 2020**

**Etudiant :** Maxime TOQUEBIAU  
**Entreprise :** Ministère des Armées  
**Tuteur de stage :** M. SYLVAIN  
**Téléphone du tuteur :** 01 56 06 08 46

**Majeure :** Big Data & Analytics

**@ du tuteur :** prospection.stage02@intradef.gouv.fr

<b>Maîtrise des domaines scientifiques et techniques :</b> <ul style="list-style-type: none"> <li>Capacité d'analyse/compréhension des problèmes</li> <li>Mise en œuvre de ses connaissances</li> <li>Aptitude à acquérir de nouvelles connaissances (formation ou autoformation)</li> </ul>	9 /10
<b>Maîtrise des méthodes et des outils de l'ingénieur :</b> <ul style="list-style-type: none"> <li>Méthodologie/organisation du travail, gestion de projet : développement d'un outil ou d'une méthodologie</li> <li>Synthèse et communication des résultats, maîtrise des outils de communication</li> </ul>	7 /10
<b>Conduite de l'action et prise de décision :</b> <ul style="list-style-type: none"> <li>Réalisation des objectifs, qualité du travail réalisé</li> <li>Autonomie/initiative/créativité/ouverture</li> <li>Respects des procédures (qualité, sécurité, santé...)</li> </ul>	8 /10
<b>Intégration dans une organisation et capacité d'animation :</b> <ul style="list-style-type: none"> <li>Capacité à s'intégrer dans une équipe : exprimer ses attentes, donner son point de vue, sens de l'écoute, accepter la critique et se remettre en cause</li> <li>Communication sur ses activités et aptitude à rendre compte (réunion, relation client...)</li> <li>Prise en compte des enjeux métiers et économiques</li> </ul>	4 /5
<b>Respect des valeurs sociétales, sociales et environnementales :</b> <ul style="list-style-type: none"> <li>Appropriation des valeurs, des codes, et de la culture de l'équipe et de l'organisation</li> <li>Comportement éthique</li> </ul>	4 /5
<b>TOTAL</b>	<b>32 /40</b>
<b>Appréciation globale sur le stage et observations</b> M. Toquebiau s'est bien investi dans son stage. Il a réalisé un état de l'art de qualité sur le sujet de la génération de dialogue et livré un prototype fonctionnel qui donne des résultats prometteurs. De plus, la présentation orale de son stage a été très appréciée. Autonome, M. Toquebiau s'est également bien adapté à la situation particulière liée au confinement, durant lequel il a avancé avec sérieux dans ses travaux. Il a fait preuve de curiosité scientifique en réalisant un état de l'art très complet, qui aurait cependant pu être plus restreint, afin de passer plus de temps sur l'implémentation du prototype et explorer l'ensemble des pistes qu'il avait proposées.	<b>TOTAL</b>  <b>16 /20</b>

A Paris le 06 Août 2020  
Signature du Tuteur de stage (obligatoire)




## **Remerciements**

Avant de commencer ce rapport, j'aimerais remercier les personnes qui ont participé à son bon déroulement. Je remercie donc en premier lieu mes deux tuteurs de stage pour leur suivi et leurs conseils pendant toute la durée de mon stage, et pour la relecture de ce rapport.

Je remercie également mon ami Antoine, étudiant à l'ECE et stagiaire au ministère des Armées, pour m'avoir accompagné pendant mon stage et aidé pour l'écriture de ce rapport.

## **Note aux lecteurs :**

Ce rapport est long et très riche en termes techniques. J'ai donc ajouté un lexique qui regroupe les différents acronymes, les termes anglais que je n'ai pas pu traduire, et les notions techniques clés à la fin du rapport.

# Sommaire

Introduction .....	6
A. Cahier des charges .....	7
B. Présentation du contexte de la mission .....	8
C. Politique de responsabilité sociétale du ministère des Armées .....	10
D. Description du travail réalisé : .....	14
1. Introduction .....	14
2. Sous-tâches de la NLG .....	15
3. Fondamentaux des approches neuronales en NLG .....	17
3.1 Réseaux de neurones récurrents .....	17
3.2 Architectures Seq2Seq .....	21
3.3 Transformers .....	23
4. État de l'art .....	27
4.1 <i>Transfer Learning</i> .....	27
4.2 Apport de connaissances au modèle .....	31
4.3 Apprentissage par renforcement .....	35
4.4 <i>Adversarial Learning</i> .....	36
4.5 Conclusion et tendances de la recherche en NLG .....	38
5. Réalisation d'un chat-bot .....	40
5.1 Définition de la tâche .....	40
5.2 Approche choisie .....	41
5.3 Collecte de données .....	42
5.4 Entraînement .....	44
5.5 Interaction avec le chat-bot .....	44
5.6 Possibles développements futurs .....	45
E. Bilan et conclusion .....	47
Lexique .....	49
Ressources .....	51
Annexes .....	57

# Introduction

Pour mon stage de fin d'études j'ai décidé de réaliser un stage de *data scientist*. Ce choix s'est fait en accord avec mes études réalisées dans la majeure *Big Data & Analytics*, ainsi que mon dernier stage de consultant en *Data Science* ("science des données" en français). Mon but était donc de trouver une mission intéressante qui puisse me faire gagner en expérience dans le milieu du *Deep Learning* (DL, ou "apprentissage profond" en français). Il m'était également très important de trouver une mission qui a du sens par rapport à mes valeurs ou mes centres d'intérêts. Il m'importe peu de développer des solutions pour faire gagner le plus d'argent possible à des entreprises qui n'aspirent à rien d'autre que le profit. Ces critères étaient d'autant plus importants que, ce stage étant mon dernier, j'avais de grandes chances d'être embauché à son issue.

J'étais donc très motivé par les offres de stage proposées par le ministère des Armées aux étudiants de l'ECE. D'abord parce que travailler pour l'État français est une mission fort honorable, mais aussi parce que les nombreux stages proposés traitaient tous de sujets très intéressants. J'ai donc postulé à certains d'entre eux et ai été convoqué pour passer un entretien pour le stage qui fait l'objet de ce rapport. Celui-ci avait pour intitulé "Génération de texte". Il avait pour but d'étudier l'état de l'art des techniques de génération automatique de texte, notamment dans le cadre d'une conversation. C'était le stage qui m'attirait le plus, car il nécessitait une étude approfondie des techniques de DL pour la génération de texte et l'utilisation de ces techniques dans un cas pratique. J'étais donc content d'être choisi pour ce poste dont le cadre et le sujet étaient tous deux grisants.

Pour travailler au ministère des Armées, plusieurs contraintes de sécurité doivent être respectées. D'abord, une enquête d'une durée d'à peu près deux semaines est réalisée sur tous les potentiels nouveaux stagiaires. Elle a pour but de vérifier la bonne foi de toutes les personnes voulant entrer au sein du ministère. Ensuite, on nous a rapidement demandé d'être discrets par rapport à nos activités professionnelles. Les travaux réalisés au ministère des Armées traitent de sujets de sécurité potentiellement très sensibles, il faut donc s'assurer qu'ils restent secrets. De même, l'organisation interne de l'administration se doit de rester confidentielle. La politique par rapport aux stagiaires est que leur travail n'est pas confidentiel. Je n'ai donc pas eu à signer de "demande de confidentialité", et je peux parler librement de tout ce sur quoi j'ai travaillé pendant mon stage. Néanmoins, il me sera interdit de parler de tout travail réalisé par d'autres personnes que moi. Je ne pourrai donc pas expliquer en détail le contexte de ma mission. La finalité de celle-ci est d'être intégrée à des développements en cours. On m'a donc rapidement expliqué l'objectif de ces développements, mais je n'aurai pas le droit d'en parler.

Dans ce rapport, j'exposerai d'abord le cahier des charges de ce stage et présenterai le contexte de ma mission. Nous verrons ensuite une présentation de la politique de responsabilité sociétale du ministère. Ensuite, nous rentrerons dans le vif du sujet, avec une description complète du travail réalisé durant mon stage. Enfin, je dresserai un bilan de mon travail et de mon expérience au sein du ministère des Armées. Nous verrons quel est l'apport de cette mission pour le ministère, et comment elle m'a amené à repenser mon avenir professionnel.

## A. Cahier des charges

La nature et la quantité de données exploitées par le ministère des Armées sont telles que les experts ne peuvent analyser l'intégralité de ces données sans l'assistance de traitements automatiques pour filtrer ou enrichir les contenus les plus pertinents. En particulier, le ministère des Armées doit traiter des données textuelles hétérogènes et multilingues. Parmi les traitements développés, certains nécessitent de générer du texte pour répondre à différentes tâches comme la traduction automatique, la synthèse de texte ou la modélisation de conversations. Ces tâches, ainsi que la génération de texte dans son ensemble, seront présentées plus en détail dans la Partie D.

Dans ce cadre, le ministère des Armées souhaite étudier la problématique de la génération automatique de texte, et en particulier la création de **systèmes conversationnels**. Ces modèles, aussi appelés chat-bots, ont pour but de comprendre la construction d'une conversation et générer des déclarations en réponse à d'autres agents, tel un être humain participant à un dialogue. Ils sont souvent utilisés dans le cadre de conversations avec des êtres humains pour les assister à la résolution d'un problème ou à l'exécution d'une commande particulière. Néanmoins, tous les systèmes conversationnels n'ont pas une tâche précise. L'objectif primordial d'un chat-bot est d'abord de ressembler le plus possible à un être humain. Une partie de la recherche est donc consacrée aux chat-bots dit "**open-domain**" ("domaine ouvert" en français), qui n'ont pas de but particulier dans la conversation, à part de ressembler le plus possible à un être humain, ce qui s'apparente au test de Turing.

Ma mission s'inscrit donc dans ce domaine. La première étape à réaliser est un **état de l'art** des techniques de génération de texte. Les **techniques neuronales**, c'est-à-dire celles qui utilisent des réseaux de neurones, sont de loin les plus efficaces, pour plusieurs raisons que nous détaillerons dans la suite de ce rapport. L'état de l'art sera donc centré sur ces approches. De plus, comme nous l'avons dit, le sujet principal de mon stage est les systèmes conversationnels. Nous étudierons donc en grande partie les approches traitant de génération de texte dans le cadre d'une conversation. Néanmoins, il est possible que des systèmes développés pour d'autres tâches apportent des idées intéressantes qui peuvent nous servir. Nous ne resterons donc pas cantonnés aux systèmes conversationnels.

L'objectif de cet état de l'art est de déterminer les outils disponibles pour construire un chat-bot *open-domain*. Une fois l'état de l'art terminé, nous étudierons les performances et la pertinence de différentes solutions existantes. Cela nous permettra de choisir une approche pour notre système. Il est possible que nous ayons besoin de collecter des données pour entraîner nos modèles. Enfin, la mission s'achèvera par la construction d'un prototype de système conversationnel. Toutes les étapes de cette mission sont résumées dans le planning prévisionnel de la Figure 1.

Tâche	Février	Mars	Avril	Mai	Juin	Juillet	Août
Cours bases génération de texte							
État de l'art génération de texte							
Tests systèmes existants							
Récolte de données							
Construction du prototype							
Rédaction du rapport							

Figure 1 : Planning prévisionnel des différentes tâches tout au long de mon stage.

## B. Présentation du contexte de la mission

### Ministère des Armées

Le ministère des Armées est l'administration responsable de la préparation et de la mise en œuvre de la politique de défense du pays. Il est dirigé par la ministre Florence Parly depuis le 21 juin 2017. C'est le président de la République Emmanuel Macron qui endosse le rôle de chef des armées.

Le ministère des Armées est créé en 1948 sous le nom de ministère de la Défense nationale. Il regroupe les trois états-majors de l'armée de terre, de la marine et de l'armée de l'air. Depuis sa création, il a changé plusieurs fois de nom, pour prendre le nom de ministère des Armées en 2017 sous le gouvernement d'Édouard Philippe. Aujourd'hui, il siège à l'hôtel de Brienne, dans le 7ème arrondissement de Paris, et à l'Hexagone Balard, dans le 15ème arrondissement de Paris. Le ministère employait plus de 260 000 personnes en 2018, avec un budget de 35,9 milliards d'euros en 2019. Le ministère des armées regroupe un très grand nombre d'organismes dont les principaux sont :

- l'État-Major des Armées,
- le Secrétariat Général pour l'Administration,
- la Direction Générale de l'Armement,
- l'Armée de terre,
- l'Armée de l'air,
- la Marine.

### Missions et enjeux

Afin d'assurer la défense et la sécurité nationale, le ministère des Armées définit **cinq grandes fonctions stratégiques** : connaître et anticiper, prévenir, dissuader, protéger et intervenir. Le rôle du ministère est donc d'abord de protéger le territoire, la population et les intérêts français. Néanmoins, il répond également à d'autres missions dans le cadre d'accords internationaux, comme avec l'OTAN ou l'Europe de la défense. Enfin, le ministère est impliqué dans des missions de service public. Ses moyens humains et matériels soutiennent les actions d'autres ministères sur le territoire national ou à l'étranger.

L'organisation et les missions de la défense nationale ont connu une profonde transformation ces dernières années. L'imprévisibilité de l'environnement international est source de risques et de menaces potentielles. Le ministère des Armées doit tenir compte de ces nouveaux facteurs d'instabilité dans ses processus et de leurs éventuelles implications pour les nouvelles zones de crise et de tension internationales. Face à ces enjeux, le ministère entreprend de grandes politiques de modernisation de son organisation et de recherche et développement.

Dans ce contexte, les nouvelles technologies de l'information et de la communication jouent évidemment un rôle décisif. Le "**cyberespace**", maillage globalisé de systèmes informatiques, est un nouveau champ de bataille où les combats se livrent à coup de perturbation des infrastructures, corruption de données ou encore désinformation. Dans cet espace, le ministère des Armées doit se positionner comme garant de la sécurité des systèmes d'informations de l'État, en développant ses propres outils de défense comme d'attaque. Mais il doit aussi savoir utiliser la puissance des



nouvelles technologies comme un outil dans ses nombreuses missions que sont la défense du territoire, l'organisation de ses opérations ou encore l'espionnage.

Ma mission s'inscrit dans la réponse à ces multiples enjeux. L'**intelligence artificielle** (IA), en plus de ses atouts économiques, est sans doute le prochain outil qui révolutionnera notre manière de nous défendre et de faire la guerre. Le développement de l'IA et son contrôle forment une course à laquelle toutes les institutions gouvernementales participent. La génération de texte, et plus généralement le **traitement de langage naturel** (ou NLP pour "*Natural Language Processing*" en anglais), ne sont que des briques de cet enjeu, mais ils constituent en eux-mêmes des outils de marque pour répondre à de nombreux problèmes. Il est donc important pour le ministère des Armées de se positionner dans ces développements pour être capable d'anticiper et de combattre les menaces, tout en assurant son autonomie face à ses alliés et sa supériorité face à ses ennemis. L'**unité de traitement média** du ministère des Armées, dans laquelle j'exerçais, a donc pour mission d'explorer les différentes manières d'utiliser les contenus multimédias, avec les derniers algorithmes d'IA, pour répondre à différents problèmes comme l'extraction ou la visualisation d'informations.

## C. Politique de responsabilité sociétale du ministère des Armées

Le ministère des Armées, comme tout autre institution de l'État, se doit de s'imposer des objectifs ambitieux en matière de développement durable. Dans cette partie nous allons donc passer en revue ces objectifs, à travers cinq axes proposés par la norme ISO 26000 :

1. la gouvernance de l'organisation
2. les droits de l'Homme
3. les relations et conditions de travail
4. l'environnement
5. la loyauté des pratiques

Nous reprendrons les différents points définis dans le questionnaire proposé par l'ECE.

### Gouvernance de l'organisation

- L'entreprise a défini des valeurs partagées par l'ensemble des salariés et en adéquation avec les principes du développement durable :

**Oui**, le ministère des Armées a défini des défis stratégiques parmi lesquels on retrouve la consommation durable, la formation du personnel ou encore la conservation de la biodiversité.

- La stratégie est communiquée au sein de l'entreprise et est en lien avec le développement durable :

**Oui**, des représentants chargés de la communication sur le développement durable ont été nommés. Des outils de communication ont également été mis en place, notamment sur l'intranet du ministère, pour fédérer les informations et favoriser les échanges entre les chargés d'environnement et tous les autres services.

- L'entreprise a mis en place des indicateurs (sociaux, environnementaux, économiques) pour piloter son activité :

**Non**, à ma connaissance, aucun indicateur n'a été mis en place.

- L'entreprise communique ses actions et décisions de façon transparente vers l'ensemble des parties prenantes :

**Oui**, le ministère communique sur ses décisions en rapport avec le développement durable, que ce soit avec ses agents, via les chargés d'environnement, ou avec les entreprises partenaires, via des comités de concertation.

### Droits de l'Homme

- L'entreprise a bien identifié les risques de non-respect des principes énoncés dans la charte des droits de l'Homme :

**Oui**, le ministère des Armées est bien conscient des risques de non-respect des principes des droits de l'Homme, notamment en matière d'armement et de prolifération nucléaire,

et contribue, comme tous les organismes de l'État, à l'élaboration de réglementations européennes et internationales.

- L'entreprise est organisée pour faire remonter toute information en cas d'atteinte constatée aux droits de l'Homme :

**Oui**, un système de lanceur d'alertes a été mis en place par l'État en ligne pour permettre à n'importe qui de signaler une violation grave et manifeste d'une loi ou d'un règlement.

- L'entreprise a mis en place des actions en faveur de la diversité, notamment pour les personnes en situation de handicap et pour l'égalité professionnelle :

**Oui**, le ministère des Armées a mis en place des plans d'action pour traiter ces sujets. On a donc par exemple le plan "égalité des chances" qui vise notamment à féminiser les armées et à répartir les postes de responsabilité entre les hommes et les femmes ; et le plan "handicap" qui vise à embaucher des personnes en situation de handicap et à favoriser l'insertion de ces personnes dans la vie professionnelle du ministère.

- L'entreprise veille au respect des droits fondamentaux (civils, politiques, économiques et culturels) :

**Oui**, comme toute institution de l'État, ces droits sont inscrits dans la politique de l'entreprise et sont donc respectés avant tout.

## **Relations et conditions de travail**

- Les relations Employeur-Employés sont basées sur le respect des principes d'égalité et le respect des droits et devoirs de chacun :

**Oui**, comme dans toutes les institutions de l'État, les droits des employés sont respectés à la lettre et les employeurs mettent un point d'honneur à respecter ces droits que ce soit dans les missions effectuées ou les relations employeur-employés.

- L'entreprise met en place les conditions de travail appropriées aux missions et une protection sociale convenable :

**Oui**, le ministère des Armées fait en sorte d'aménager au mieux le cadre de travail pour permettre à chacun de réaliser ses missions dans les meilleures conditions possibles. Il a également mis en place une mutuelle spéciale pour les agents du ministère des Armées pour garantir une protection sociale adaptée.

- L'entreprise favorise le dialogue social avec son personnel et ses instances représentatives :

**Oui**, le ministère des Armées organise des concertations avec son personnel, notamment via des "comités techniques" qui servent de lieux de discussion et d'information entre l'administration et les organisations syndicales représentant le personnel.

- L'entreprise prend toutes les mesures nécessaires pour garantir la santé et la sécurité de son personnel :

**Oui**, la santé de son personnel militaire et civil est une mission prioritaire du ministère des Armées. Cela passe par un suivi médical de tous les agents (y compris les stagiaires) du ministère, un contrôle de l'hygiène et de la sécurité de la chaîne alimentaire, une politique de prévention des risques professionnels (troubles physiques ou psychiques, prévention vaccinale, prévention des comportements addictifs et suivi des risques cancérogènes) et une prise en charge médicale des personnes blessées en service.

- L'entreprise a mis en place une politique de gestion prévisionnelle des emplois et compétences :

**Oui**, la formation professionnelle est considérée comme un droit fondamental pour tous les personnels du ministère. Ce droit est mis en œuvre grâce à de nombreux centres de formation militaires et civils, une politique de formation continue des agents et la constitution d'une agence de reconversion à la défense qui dispose de nombreuses formations dans 7 domaines professionnels différents.

## **L'environnement**

- L'entreprise a qualifié tout type de pollution inhérent à son activité et prend les mesures nécessaires pour les prévenir et les réduire :

**Oui**, le ministère des Armées étant dépendant à 85% des énergies fossiles, la gestion des énergies est un enjeu stratégique prioritaire. La base de son action de sensibilisation passe par la connaissance de ses consommations d'énergies. Pour cela, une gouvernance globale du sujet énergétique a été mise en place au sein du ministère. Des études approfondies sont réalisées sur la consommation d'énergie et de carburants.

- L'entreprise optimise sa consommation de ressources (matières premières, consommables, énergie) :

**Oui**, le ministère a mis en place des politiques de rationalisation de la consommation d'énergie, de recours à des énergies renouvelables et d'optimisation énergétique des infrastructures et des transports.

- L'entreprise a identifié l'impact de ses activités sur le changement climatique et met en œuvre toutes les mesures pour les réduire :

**Oui**, les politiques d'étude des impacts environnementaux du ministère et d'optimisation énergétique de ses activités visent toutes à réduire au maximum l'empreinte écologique du ministère des Armées.

## **Loyauté des pratiques**

- L'entreprise a mis en place un dispositif visant à lutter contre la corruption :

**Oui**, l'Arrêté du 23 août 2018 prévoit les procédures de recueil des signalements visant à la transparence et à la lutte contre la corruption.

- L'entreprise encourage la transparence en matière de politiques publiques. Elle veille à éviter les conflits d'intérêt et les abus d'influence :

**Oui**, malgré le secret nécessaire encadrant une grande partie des opérations du ministère des Armées, celui-ci effectue des efforts de transparence notamment en matière de désarmement, avec par exemple des visites d'installations nucléaires démantelées.

- L'entreprise a mis en place des procédures garantissant le respect des droits de propriété :

**Oui**, le ministère des Armées a mis en place une politique immobilière permettant de gérer le parc immobilier du ministère et de donner accès aux personnels militaires et civils, en activité ou à la retraite, à des prêts destinés à favoriser l'acquisition d'un bien immobilier ou la réalisation de travaux.

## **Conclusion**

En tant qu'institution de l'État, le ministère des Armées se doit d'être exemplaire quant au respect des droits de ses agents et des politiques environnementales du pays. Les directives relatives à ces deux points, fixées par l'État et à l'intention des entreprises exerçant sur le sol français, se doivent d'être respectées à la lettre par toutes les instances civiles du pays. Le cas du ministère des Armées reste néanmoins spécial, du fait que sa mission est avant tout la sécurité des Français et de leurs intérêts. Cette mission peut engager certains sacrifices en matière de consommation énergétique, par exemple. Mais il est justement important de fixer clairement des directives strictes pour pouvoir respecter au maximum les valeurs du développement durable.

En tant que stagiaire au sein du ministère des Armées, il est clair que les droits fondamentaux des employés sont une priorité pour les employeurs et pour le ministère d'une manière générale. Une attention particulière est donnée à la mise en place de conditions de travail exemplaires, à la santé des employés et à leur accès au logement. Des campagnes de prévention de risques de santé sont mises en place et des logements à faibles prix sont proposés.

Sur le plan des directives environnementales, il est difficile, pour un stagiaire, de les voir en application. Il est donc dur de juger du respect de ces directives, mais force est de constater qu'il y a une volonté claire d'améliorer l'empreinte écologique des activités du ministère.

## D. Description du travail réalisé :

### Génération de texte :

#### Étude de l'état de l'art et réalisation d'un chat-bot

### 1. Introduction

La **Génération de Langage Naturel** (ou NLG pour *Natural Language Generation*) a été définie en 2000 par Reiter et Dale comme « *le sous-champ de l'IA et du NLP qui concerne la construction de systèmes informatiques permettant de produire des textes compréhensibles grâce à une **représentation non-linguistique** sous-jacente de l'information* ». Cette définition rassemble de nombreux domaines dont les principaux sont présentés dans la Figure 1.1. Nous aborderons certains d'entre eux, en approfondissant particulièrement le sujet qui motive cette étude : les **systèmes conversationnels**. Le but ultime de la NLG est de produire, quelle que soit la tâche, un texte possédant toutes les qualités (et les défauts ?) d'un texte écrit par un être humain. Depuis les années 1970, de nombreuses approches ont été explorées pour répondre au mieux à ce besoin général, ainsi qu'aux besoins spécifiques à chaque tâche. Ces approches sont similaires à ce qu'on peut trouver dans les autres domaines de la recherche en IA. On trouve d'abord des systèmes modulaires séparant la tâche en sous-problèmes résolus séparément, souvent grâce à un ensemble de règles (Gatt and Krahmer, 2018). A partir des années 1990, on assiste à un changement de paradigme vers des systèmes dits "**end-to-end**" ("de bout en bout" en français) qui cherchent à résoudre leurs tâches d'une manière plus globale, intégrant les sous-problèmes en un seul. De multiples techniques ont été développées au sein de ces systèmes : en partant de la planification grammaticale, pour aller vers des approches statistiques qui se concentrent sur l'exploitation de larges corpus de données. C'est cette dernière approche qui est majoritairement développée aujourd'hui. On y retrouve notamment les techniques neuronales, qui emploient des réseaux de neurones pour modéliser au mieux les caractéristiques des corpus. Quand ces réseaux sont "profonds", c'est-à-dire qu'ils utilisent plusieurs couches de neurones pour atteindre un haut niveau d'abstraction des données, on parle alors de *Deep Learning*. Ces techniques se sont largement imposées comme les plus efficaces, notamment grâce à leur grande faculté de généralisation.

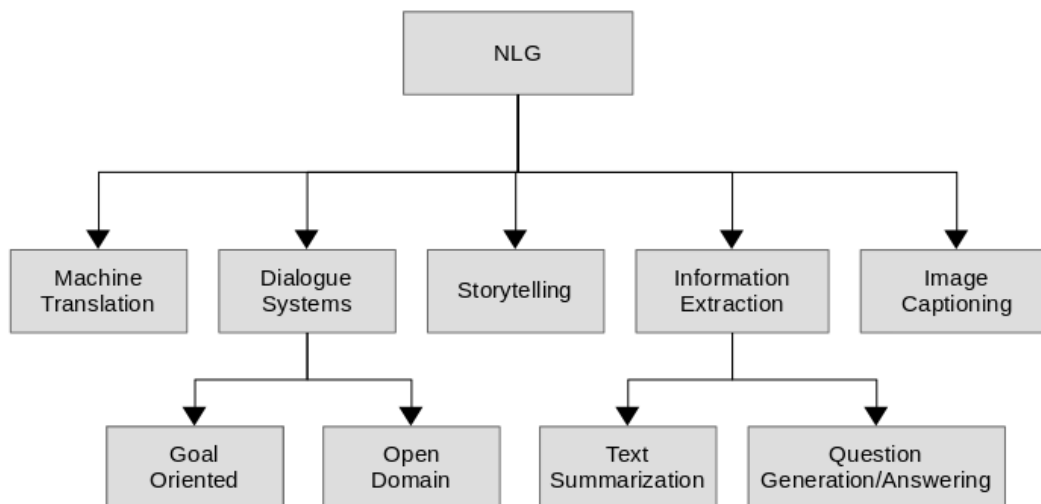


Figure 1.1 : Les différents domaines d'application de la NLG

Nous commencerons donc par une rapide étude des sous-problèmes rencontrés en NLG. Ensuite, nous détaillerons le fonctionnement des techniques que nous pourrions utiliser pour notre système conversationnel, à savoir les modèles de DL utilisés aujourd'hui. Enfin, nous expliquerons nos choix ainsi que les étapes réalisées pour construire notre chat-bot.

## 2. Sous-tâches de la NLG

Comme dit plus haut, la première approche pour générer du texte cherchait à séparer la tâche en plusieurs sous-problèmes. Le but était de définir les étapes nécessaires pour convertir des données, quelle que soit leur forme, en un texte. On trouve dans de nombreux systèmes de NLG les six étapes suivantes (Reiter and Dale, 2000) :

1. **Détermination du contenu** : Cette étape décide de quelles informations inclure dans le texte en cours de génération. En fonction de l'audience du texte généré (par exemple, est-elle constituée d'experts ou de novices) ou de son intention (cherche-t-il à informer ou à convaincre), différentes informations seront mises en avant. Certaines données d'entrée peuvent également être inutiles à l'utilisateur. Il faut donc trouver des techniques pour choisir quelles données garder dans le texte final. Parmi les différentes solutions, on trouve d'abord des systèmes basés sur des templates à remplir avec les données importantes (Goldberg et al., 1994). Plus récemment, la recherche s'est orientée vers des solutions dites "*data-driven*" (« basées sur les données » en français) pour identifier les sujets intéressants à inclure, en utilisant des techniques de classification.
2. **Structuration du texte** : Après avoir déterminé les informations à communiquer, il faut décider de l'ordre dans lequel elles seront présentées. Là encore la tâche est influencée par le domaine d'application. Dans la génération de compte-rendu d'événements sportifs, on voudra commencer par des informations générales, pour ensuite entrer progressivement dans les détails. A l'inverse, pour générer un diagnostic médical, on cherchera à exposer les faits chronologiquement. Pour répondre à ce problème, beaucoup de travaux ont utilisé la *Rhetorical Structure Theory* (RST, ou "théorie de la structure rhétorique" en français) proposée par Mann et Thompson en 1987 (Mann and Thompson, 1998). La RST définit des relations entre différentes parties de textes : des parties importantes appelées « nuclei », et des parties supplémentaires appelées "satellites". D'autres approches utilisant des algorithmes de ML ont été explorées pour structurer les informations automatiquement, avec notamment l'utilisation de l'indicateur Tau de Kendall qui note la corrélation entre plusieurs phrases (Lapata, 2006), ou encore l'apprentissage supervisé d'un critère d'association et d'ordre entre deux phrases (Bollegala et al., 2010).
3. **Agrégation de phrases** : Toutes les informations à citer ne nécessitent pas une phrase distincte. On va souvent pouvoir rassembler plusieurs informations en une phrase, rendant le résultat plus fluide et lisible. Pour ce faire, les méthodes *data-driven* sont favorisées. Par exemple, Barzilay et Lapata (2006) regardent des paires de phrases et déterminent si elles devraient être agrégées en se basant sur leur similarité.
4. **Lexicalisation** : Une fois que le contenu du texte à générer est finalisé, il faut choisir quels mots et expressions utiliser pour l'exprimer. Cette tâche est la plus

délicate car elle doit retranscrire le sens abstrait des informations dans les objets concrets que sont les mots. Cela peut s'avérer compliqué, d'abord parce qu'on trouve dans les corpus de nombreuses manières d'exprimer des informations très similaires. Ensuite, parce que dans certains cas, une simple nuance dans le vocabulaire peut changer le sens d'une phrase, voire la rendre totalement inadéquate dans le contexte du texte généré. Cette tâche de lexicalisation est donc souvent réalisée de manière spécifique au domaine d'application. Souvent, elle est effectuée en même temps que d'autres étapes comme la réalisation linguistique, expliquée plus bas, avec par exemple les templates de phrases à remplir évoqués plus haut (Goldberg et al., 1994).

5. **Génération de représentations référentielles** : Notre langage est composé de mots qui servent de représentations pour les idées ou informations que nous voulons communiquer. Une représentation référentielle est un mot, ou groupe de mots, qui représente un objet, concret ou abstrait. Une étape de la génération de texte correspond donc à trouver des représentations référentielles pour des personnes, des lieux ou des événements.
6. **Réalisation linguistique** : Enfin, une fois que tous les mots et expressions à utiliser sont déterminés, il faut les combiner et les arranger pour former des phrases linguistiquement correctes. Pour cela, la première approche est celle des templates de phrases (Goldberg et al., 1994). Ceux-ci sont tous créés par des humains. Par conséquent, ils permettent de retranscrire efficacement le message voulu mais limitent grandement la variété des phrases générées. Une alternative aux templates se trouve dans les systèmes basés sur la grammaire qui forment les phrases grâce à un ensemble de règles codées à la main (Bateman, 1997). Leur avantage est qu'ils ne dépendent pas du domaine d'application. Cependant, ils nécessitent des entrées très détaillées, ce qui les rend difficiles à intégrer au sein d'autres projets (Gatt and Krahmer, 2018). Enfin, d'autres méthodes ont des approches plus statistiques du problème. En se basant sur une étude de larges corpus de données, l'information statistique est utilisée dans des modèles de langue pour filtrer les résultats d'un système basé sur la grammaire (Langkilde and Knight, 1998 ; White and Rajkumar, 2009).

Ces six sous-problèmes résument donc bien la tâche de NLG. Il est important de noter que tous les systèmes n'approchent pas ces étapes de la même manière. Souvent, leur ordre est changé ou elles sont regroupées pour être réalisées en même temps dans un même module.

On observe une tendance dans la recherche vers un abandon des approches basées sur des systèmes de règles codées à la main, pour privilégier les approches stochastiques basées sur l'étude statistique de larges corpus de données. De même, les approches indépendantes du domaine d'application sont de plus en plus développées. Dans ce sens, les techniques du DL ont progressivement fait leur apparition en NLG. Celles-ci permettent de modéliser nos langages en tirant profit de corpus de plus en plus riches et nombreux. De plus, elles permettent d'implémenter des solutions « *end-to-end* » : on donne des données en entrée (texte, données numériques, images...) et le système génère automatiquement le texte de sortie. Toutes les étapes citées plus haut sont donc réalisées dans le même module, quasiment simultanément. Dans la prochaine partie, nous nous pencherons sur ces approches neuronales en partant de leurs bases, pour aller jusqu'aux techniques à la pointe de l'état de l'art.



### 3. Fondamentaux des approches neuronales en NLG

Avant d'étudier l'état de l'art des techniques neuronales en NLG, commençons par expliquer les éléments sur lesquels se basent ces techniques. Ces éléments ne sont pas très nombreux, mais il est nécessaire de comprendre leur utilité et leur fonctionnement car ils se retrouvent dans toutes les approches que nous étudierons dans la Partie 4.

#### 3.1 Réseaux de neurones récurrents

##### 3.1.1 RNN

Le premier bloc fondamental des approches neuronales en NLG est le **réseau de neurones récurrents** (RNN, pour "*Recurrent Neural Network*" en anglais). Les RNN sont des réseaux de neurones particulièrement efficaces pour traiter des données séquentielles. On les retrouve donc dans la prédiction sur des séries de données temporelles, comme le cours d'une action ou des données météorologiques, mais aussi logiquement dans le traitement du langage naturel, celui-ci étant constitué de séquences de signes, mots et phrases qui se suivent. Les RNN sont donc utilisés dans la plupart des tâches du NLP, allant de la reconnaissance automatique de la parole à la traduction automatique, en passant par la description automatique d'images.

Pour traiter efficacement des données séquentielles, les RNN s'inspirent de leurs créateurs en faisant persister l'information qu'ils traitent. Tout comme les êtres humains ne recommencent pas leur réflexion de zéro à chaque instant, les RNN utilisent le résultat de leurs traitements passés pour produire le résultat au temps  $t$ . Pour ce faire, le résultat du calcul d'un neurone récurrent est passé comme entrée de ce même neurone pour réaliser le prochain calcul. La Figure 3.1 montre un schéma d'un bloc récurrent. On voit sur la représentation de gauche la boucle qui forme une **liaison récurrente** permettant à l'information d'être passée d'une étape du modèle à la suivante. La représentation de droite montre les états successifs du modèle. Au temps  $t$ , le bloc prend en entrée l'**input** (données d'entrée) du modèle  $x_t$  ainsi que "l'état caché" ("*hidden state*" en anglais) précédent du bloc  $h_{t-1}$ , et produit un **output** (résultats des calculs réalisés par le modèle)  $y_t$ . L'état caché est une représentation numérique de l'état du modèle, calculée par celui-ci à partir de  $x_t$  et  $h_{t-1}$ . Il contient donc de l'information sur plusieurs états précédents du modèle, et donc sur plusieurs entrées précédentes du modèle.

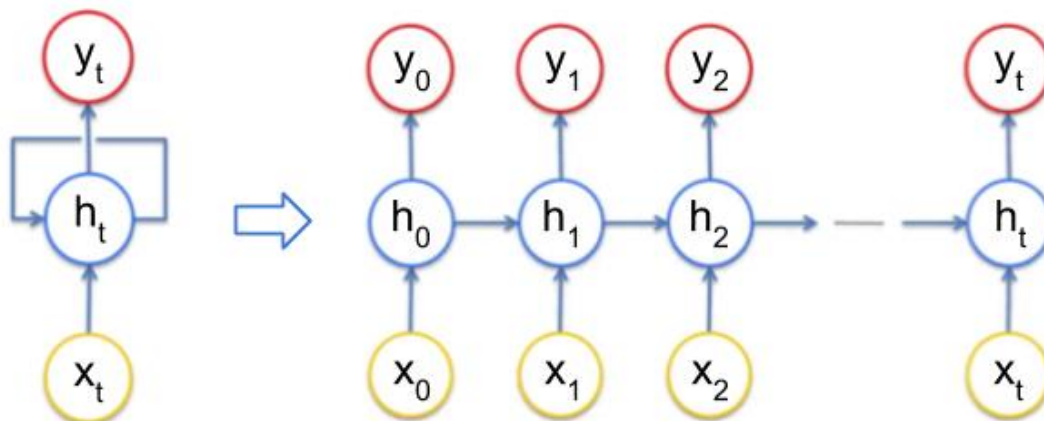


Figure 3.1 : Schéma d'un RNN, avec une boucle à gauche, déplié à droite. [1]

C'est cette nature en chaîne qui lie intimement les RNN aux séquences de données et qui permet de faire persister de l'information passée dans nos calculs. Elle permet d'opérer sur des séquences de données : séquences en entrée, en sortie ou, dans le cas le plus fréquent, les deux. La Figure 3.2 présente ces différentes formes de modèle. Tout à gauche, on a un réseau de neurones traditionnel qui produit un objet en sortie, pour un objet en entrée. À sa droite, on trouve plusieurs configurations permises par les RNN. La première prend un objet en entrée et produit, de manière séquentielle, une séquence d'objets en sortie. Un exemple d'application utilisant ce type de modèle serait la description automatique d'une image : on prend en entrée une représentation de l'image et on produit en sortie une séquence de mots. La deuxième configuration prend une séquence d'objets en entrée et produit un objet en sortie. Elle correspondrait par exemple à la classification de sentiment d'un texte. Les deux dernières configurations illustrent les modèles dits **Seq2Seq**, pour "*Sequence-to-Sequence*", qui produisent une séquence à partir d'une autre séquence, mais de deux manières différentes. La première encode d'abord tous les objets de la séquence d'entrée dans une représentation numérique, pour ensuite décoder cette représentation en produisant une séquence d'objets. Elle correspond par exemple à de la traduction automatique, où le modèle lit une phrase en anglais puis la traduit en français. Enfin, la dernière configuration génère un objet pour chaque objet de la séquence d'entrée, de manière synchronisée. Un exemple serait un modèle classifiant chaque image d'une vidéo. Il est important de noter que toutes ces configurations permettent de traiter des séquences de tailles variables, autant en entrée qu'en sortie. C'est un grand atout car la taille des séquences étudiées (en entrée) ou générées (en sortie) varie systématiquement.

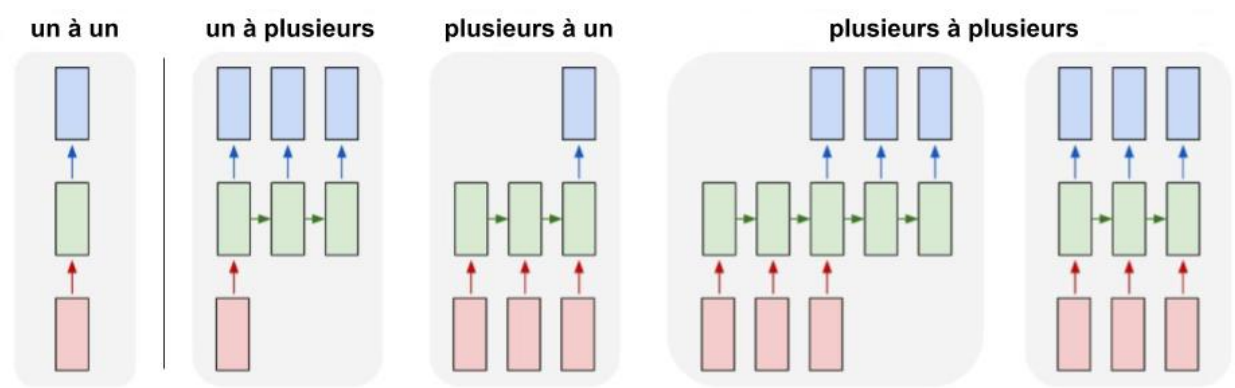


Figure 3.2 : Tout à gauche une représentation d'un réseau de neurones traditionnel, sans liaison récurrente. À droite, différentes configurations opérant sur des séquences, en entrée et/ou en sortie, possibles grâce aux RNN. [2]

### 3.1.2 Problème et solutions de la dépendance à long terme

Bien qu'en théorie, les RNN classiques sont capables de mémoriser de l'information à long terme, en pratique ce n'est pas le cas (Bengio et al., 1994). Cela est dû à l'algorithme utilisé pour entraîner ces modèles. L'entraînement d'un réseau de neurones comporte trois grandes étapes. Tout d'abord, on effectue la "**propagation avant**" pendant laquelle le modèle calcule une prédiction à partir des données en entrée. Ensuite, on compare la prédiction aux données réelles en utilisant une fonction de coût. Cette fonction de coût produit une valeur d'erreur qui juge de la qualité de la prédiction du modèle. Enfin, on utilise cette valeur pour modifier les paramètres du modèle afin de

réaliser, plus tard, des prédictions plus proches de la réalité. On dit qu'on "**rétro-propage l'erreur**" dans le modèle. Pour ce faire, on calcule les **gradients** de chaque paramètre du réseau. Le gradient d'un paramètre est la valeur utilisée pour ajuster ce paramètre, ce qui permet au modèle d'apprendre à minimiser ses erreurs. Plus le gradient est grand, plus l'ajustement est grand, et vice versa. C'est justement là où se situe le problème. Dans un RNN, on doit rétro-propager l'erreur en arrière "dans le temps", c'est-à-dire ajuster les paramètres en fonction des entrées passées du modèles. Or, en pratique, plus on remonte dans le temps plus les valeurs des gradients diminuent, de manière exponentielle. C'est ce qu'on appelle le problème de la "**disparition du gradient**". Comme des petits gradients signifient des petits ajustements sur les paramètres du modèle, ceux-ci sont peu ajustés en rapport avec les entrées moins récentes. Les RNN sont donc incapables d'apprendre des dépendances à long terme.

Une solution apportée pour résoudre ce problème a été de modifier l'architecture du modèle. Les réseaux "**Long Short-Term Memory**" (LSTM) sont un type particulier de RNN capable d'apprendre des dépendances à long terme. Ils ont été introduits par Hochreiter et Schmidhuber (1997) et ont été améliorés et popularisés dans de nombreux travaux ultérieurs. Les LSTM sont explicitement conçus pour éviter le problème de la disparition des gradients. Ils sont donc capables de se souvenir d'informations pendant de longues périodes. Les LSTM fonctionnent extrêmement bien pour une grande variété de problèmes et sont maintenant largement utilisés.

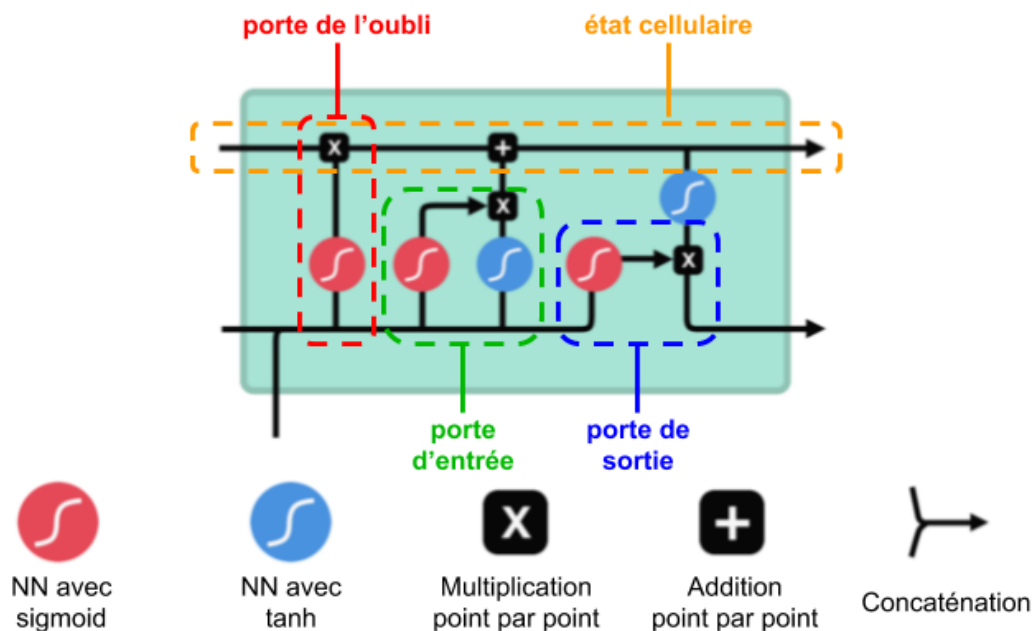


Figure 3.3 : Schéma d'un module LSTM. [3]

Les LSTM fonctionnent avec des mécanismes internes appelés "portes" qui servent à réguler les flux d'information. On peut voir sur la Figure 3.3 les concepts clés qui rendent les LSTM si efficaces. "L'**état cellulaire**" agit comme une autoroute de transport qui transfère les informations tout au long de la chaîne de traitement de la séquence. On peut le considérer comme la mémoire du réseau. L'état cellulaire peut, en théorie, transporter des informations des premiers éléments de la séquence jusqu'aux

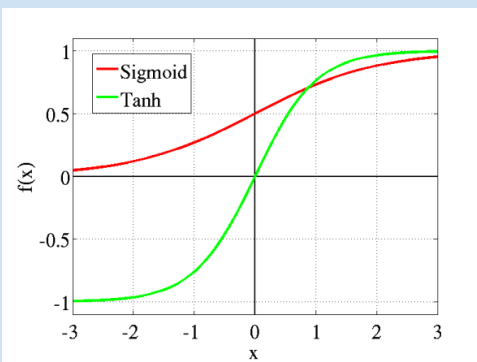
derniers. Pendant le traitement de la séquence, des informations sont ajoutées ou retirées à l'état cellulaire par des portes. Les portes sont différents réseaux de neurones qui décident des informations autorisées à persister dans l'état cellulaire. Elles peuvent apprendre quelles informations sont pertinentes à conserver ou à oublier pendant l'entraînement.

Premièrement, nous avons la “**porte de l'oubli**” qui décide de quelles informations doivent être jetées ou conservées dans l'état cellulaire. Les informations de l'état caché précédent et de l'entrée actuelle sont concaténées et passées dans un réseau de neurones avec comme activation la fonction sigmoïde (voir Note 3.1). L'état cellulaire est ensuite multiplié point par point au résultat. La sigmoïde est utile pour mettre à jour ou oublier des données : tout nombre multiplié par 0 sera oublié et tout nombre multiplié par 1 sera conservé. Le réseau de neurones apprend quelles informations sont importantes à oublier ou conserver.

Nous avons ensuite la “**porte d'entrée**” pour mettre à jour l'état cellulaire. D'abord, comme dans la porte de l'oubli, l'état caché précédent et l'entrée actuelle sont concaténés et passés dans un réseau de neurones avec comme activation une sigmoïde. Il décide des valeurs qui seront mises à jour en écrasant les valeurs entre 0 et 1. Une valeur proche de 0 signifie que l'information correspondante n'est pas importante à conserver. À l'inverse, on utilisera les informations dont la valeur, selon ce premier réseau de neurones, est proche de 1 pour mettre à jour l'état cellulaire. Afin de réguler les valeurs des informations dans le réseau, elles sont également passées dans un réseau avec comme activation la fonction tanh, qui écrase les valeurs entre -1 et 1 (voir Note 3.1). Les résultats des deux réseaux sont multipliés pour définir quelles informations seront ensuite ajoutées, par une addition simple, à l'état cellulaire. Le nouvel état cellulaire est alors calculé.

Enfin, nous avons la “**porte de sortie**” qui calcule l'état caché suivant. C'est lui qui sera utilisé pour réaliser nos prédictions. Il a donc besoin d'informations des entrées précédentes, contenues dans l'état cellulaire, du dernier état caché et de l'entrée actuelle, contenues dans l'entrée du module. Les premières sont passées dans un réseau à tanh, et les deuxièmes par un réseau à sigmoïde. Les résultats des deux réseaux sont multipliés pour produire le nouvel état caché. Le nouvel état cellulaire et le nouvel état caché peuvent donc être passés à l'étape suivante.

### Note 3.1 : fonctions d'activation sigmoïde et tanh



Les fonctions d'activation servent à contrôler les valeurs de sortie d'une couche de neurones, par exemple (voir figure ci-contre).

La fonction d'activation **sigmoïde** “écrase” les valeurs qu'on lui donne dans l'intervalle  $[0, 1]$ . Plus l'entrée est grande négativement, plus la sortie sera proche de 0. Plus l'entrée est grande positivement, plus la sortie sera proche de 1.

La fonction d'activation tangente hyperbolique, ou **tanh** “écrase” les valeurs qu'on lui donne dans l'intervalle  $[-1, 1]$ . Plus l'entrée est grande négativement, plus la sortie sera proche de -1. Plus l'entrée est grande positivement, plus la sortie sera proche de 1.

Grâce à cette architecture, les gradients ne disparaissent plus car ils peuvent être transmis via l'état cellulaire. C'est ce qui permet aux LSTM de développer des dépendances à certaines informations sur le long terme.

### 3.2 Architectures Seq2Seq

Les réseaux récurrents que nous avons étudié dans la partie précédente sont à la base des différentes architectures des modèles permettant de générer du texte. Comme nous l'avons vu dans la Partie 3.1.1, les réseaux récurrents permettent, dans certaines configurations, de produire des séquences de données à partir d'autres séquences qu'ils prennent en entrée. Ce sont ces modèles, dits Seq2Seq, qui nous intéressent le plus. En effet, notre tâche finale nécessite ce genre de modèle : dans une conversation, les participants s'échangent des phrases, qui sont elles-mêmes des séquences de mots. □ On veut donc être capable de prendre une séquence de mots, appelée "**requête**", en entrée et de produire une autre séquence de mots, qui sera notre **réponse**. Pour ce faire, deux architectures sont possibles le modèle de langue ou l'encodeur-décodeur.

#### 3.2.1 Modèle de langue

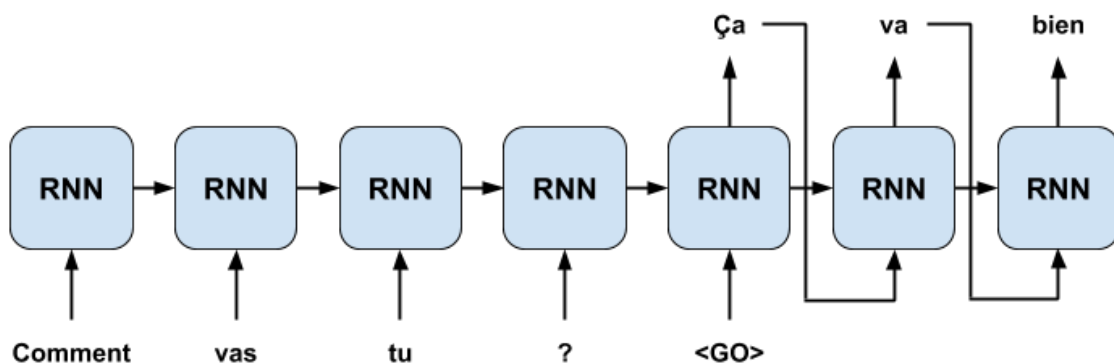
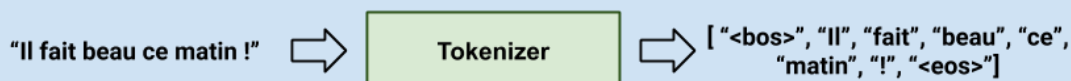


Figure 3.4 : Schéma d'un modèle de langue dans une configuration Seq2Seq.

Modéliser une langue correspond à prédire le prochain mot, sachant les mots passés. Un **modèle de langue** est donc un système qui associe des probabilités à des mots ou séquences de mots. L'idée est d'observer des corpus de texte et en tirer une information statistique sur les mots qu'ils contiennent. Avec cette information, le modèle est capable de prendre une séquence de mots et d'estimer la probabilité, pour chaque mot de son vocabulaire, d'être le prochain. Dans le cadre de la NLG, on peut donc utiliser un modèle de langue pour générer une séquence de mots, en prenant successivement le mot avec la meilleure probabilité de la distribution jusqu'à avoir une phrase. Un modèle de langue nous permet donc d'avoir un modèle Seq2Seq, en lisant d'abord la séquence d'entrée, puis en générant une séquence à la suite. Ce processus est illustré dans la Figure 3.4. On voit que la réponse est générée comme la suite de la phrase d'entrée. Le RNN traite successivement tous les "morceaux" de la séquence d'entrée, qu'on appelle **tokens** (voir Note 3.2). Pendant cette phase, la sortie du modèle de langue ne nous intéresse pas, on veut juste lire la phrase d'entrée. Le *token* spécial "<GO>" indique au modèle que la séquence d'entrée est terminée. À partir de ce moment, on regarde la

sortie du RNN à chaque étape pour générer un nouveau mot. Les mots générés servent d'entrée pour le modèle à l'étape suivante.

### Note 3.2 : *tokens*



Pour pouvoir traiter des phrases avec nos modèles, celles-ci sont d'abord découpées en morceaux appelés **tokens** ("jetons" en français). En fonction du modèle, les *tokens* peuvent être des morceaux de mots, des mots, des signes de ponctuation ou encore des *tokens* spéciaux qui servent à indiquer au modèle certaines choses. On trouve notamment très souvent les *tokens* "<BOS>" et "<EOS>", qui servent respectivement à indiquer le début d'une séquence ("*beginning of string*" en anglais) et la fin d'une séquence ("*end of string*" en anglais).

### 3.2.2 Encodeur-décodeur

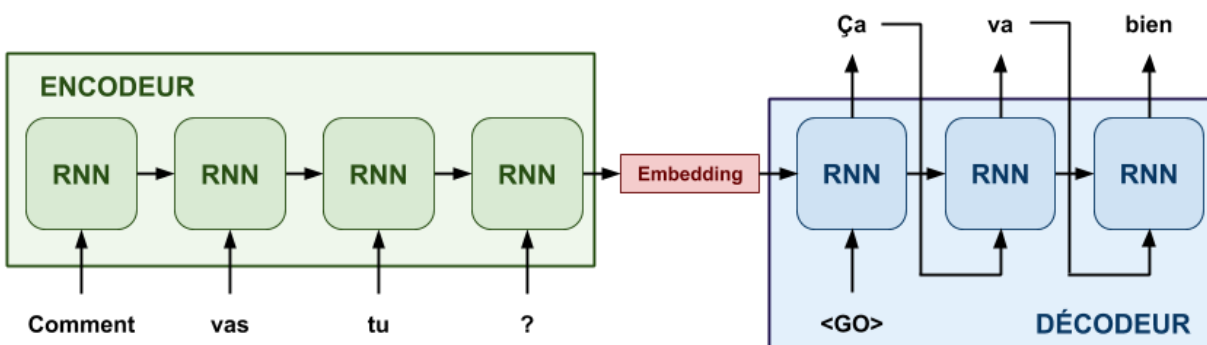
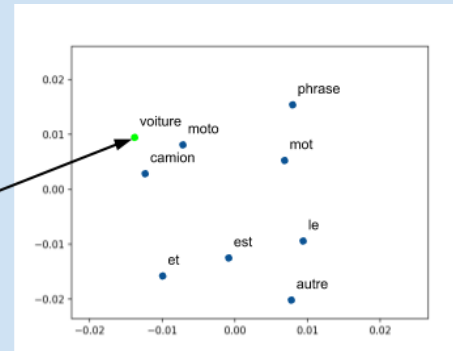
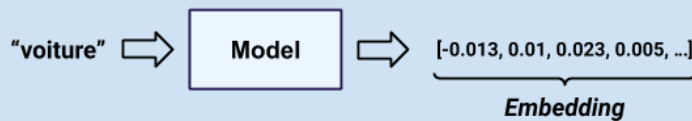


Figure 3.5 : Schéma d'un encodeur-décodeur simple utilisant deux RNN.

Basé sur le modèle de langue, l'**encodeur-décodeur** ajoute une couche d'abstraction des données en encodant d'abord la séquence d'entrée en une représentation numérique, avant de la décoder pour générer une nouvelle séquence (voir Figure 3.5). Cette représentation est appelée **embedding** (voir Note 3.3). C'est un vecteur de nombres réels qui est censé représenter au mieux la séquence d'entrée traitée par l'encodeur. À partir de cet *embedding*, le décodeur génère une séquence de mots, à la manière d'un modèle de langue. Dans la Figure 3.5, l'*embedding* est l'état caché de l'encodeur, après avoir traité le dernier *token* de la séquence d'entrée. On comprend donc que le modèle de langue présenté dans la Figure 3.4 est en fait une version très simplifiée d'un encodeur-décodeur, avec un seul RNN qui s'occupe de lire la séquence d'entrée et de l'encoder, puis de la décoder en générant la réponse. La différence entre modèle de langue et encodeur-décodeur réside donc dans le fait que l'encodeur-décodeur comporte deux modèles distincts : un pour encoder la séquence d'entrée, un autre pour générer la séquence de sortie. Dans la Figure 3.4, l'encodeur et le décodeur utilisent tous les deux un RNN pour réaliser leur tâche, mais on pourrait avoir des LSTM ou, on le verra plus tard, d'autres sortes de modèles à la place.



### Note 3.3 : *embedding*



Un **embedding** ("imbrication" en français) est une représentation d'une variable discrète, sous la forme d'un vecteur de nombres continus. Dans le contexte des réseaux de neurones, ces *embeddings* servent à représenter les objets traités : variables catégorielles, images, mots, etc.

## 3.3 Transformers

En 2017, le papier "*Attention is all you need*" (Vaswani et al.) présente une nouvelle architecture appelée **Transformer**. Celle-ci est également de type encodeur-décodeur, mais elle n'implique pas de réseau récurrent. Comme le titre du papier l'indique, le Transformer est basé sur un mécanisme appelé **attention**. Celui-ci a été introduit en 2014 (Bahdanau et al.) pour améliorer les modèles Seq2Seq. Il leur permet de se concentrer sur les parties les plus pertinentes de la séquence d'entrée pendant la génération. Les Transformers se sont rapidement imposés et sont en voie de remplacer leurs prédécesseurs récurrents, grâce à une plus grande rapidité pendant l'entraînement, la possibilité d'agrandir leur taille facilement pour avoir de meilleurs résultats et, simplement, une plus grande efficacité sur toutes les tâches du NLP. Dans cette partie nous allons essayer de comprendre les raisons de ces atouts, en reprenant d'abord le concept d'attention, pour ensuite étudier l'architecture du Transformer.

### 3.3.1 Mécanisme d'attention

Le principe du mécanisme d'attention est de pouvoir se concentrer sur une certaine partie de l'*input* pour produire l'*output*. Cela peut être intéressant dans de nombreuses tâches de NLP. Par exemple, en traduction automatique, tous les mots de la phrase à traduire n'auront pas forcément de l'importance pour choisir certains mots de la phrase traduite. Dans ce genre de cas, on peut donc ajouter le mécanisme d'attention à l'architecture encodeur-décodeur. Le principe est d'utiliser tous les états cachés calculés par l'encodeur à chaque étape et d'essayer de trouver ceux qui nous intéressent le plus pour produire un nouveau mot à une étape donnée. On peut visualiser ce processus dans la Figure 3.6, où on traduit une phrase de l'anglais au français. Au moment de générer le mot "faites", le modèle ne se base plus que sur le dernier état caché du décodeur, mais aussi sur des scores  $\alpha$  calculés en fonction des états cachés de l'encodeur, et du dernier état caché du décodeur. Cela permet de se concentrer plus sur le verbe "do", en dernière position, que sur le premier mot "what".

Ce mécanisme d'attention est assez intuitif, car le raisonnement est assez proche de celui qu'un être humain pourrait avoir pour traduire une phrase, par exemple. En outre, en plus d'augmenter les performances des modèles récurrents classiques, il permet également de mieux interpréter les résultats des modèles. Dans la Figure 3.7, on peut

voir une visualisation des scores d'attention calculés pour chaque *token*, dans le cadre de la traduction d'une phrase. On voit très bien la logique derrière le mécanisme d'attention se dessiner. Le français et l'anglais étant des langues très proches dans leur syntaxe, la plupart des mots font attention au mot placé au même endroit dans l'autre phrase, ce qui crée cette diagonale plus claire. En revanche, on peut clairement visualiser la différence d'ordre des mots dans un groupe nominal entre les deux langues. Le groupe "zone économique européenne" correspond au groupe "*European Economic Area*". La position des adjectifs par rapport aux noms dans les deux langues est inversée, et le mécanisme d'attention a clairement saisi cette règle. On peut également voir que le déterminant "la" fait attention à deux mots : "*the*" et "*Area*". Le déterminant anglais ne montrant pas d'information sur le genre, le modèle a dû également se concentrer sur le nom qui suit pour pouvoir traduire de manière adéquate.

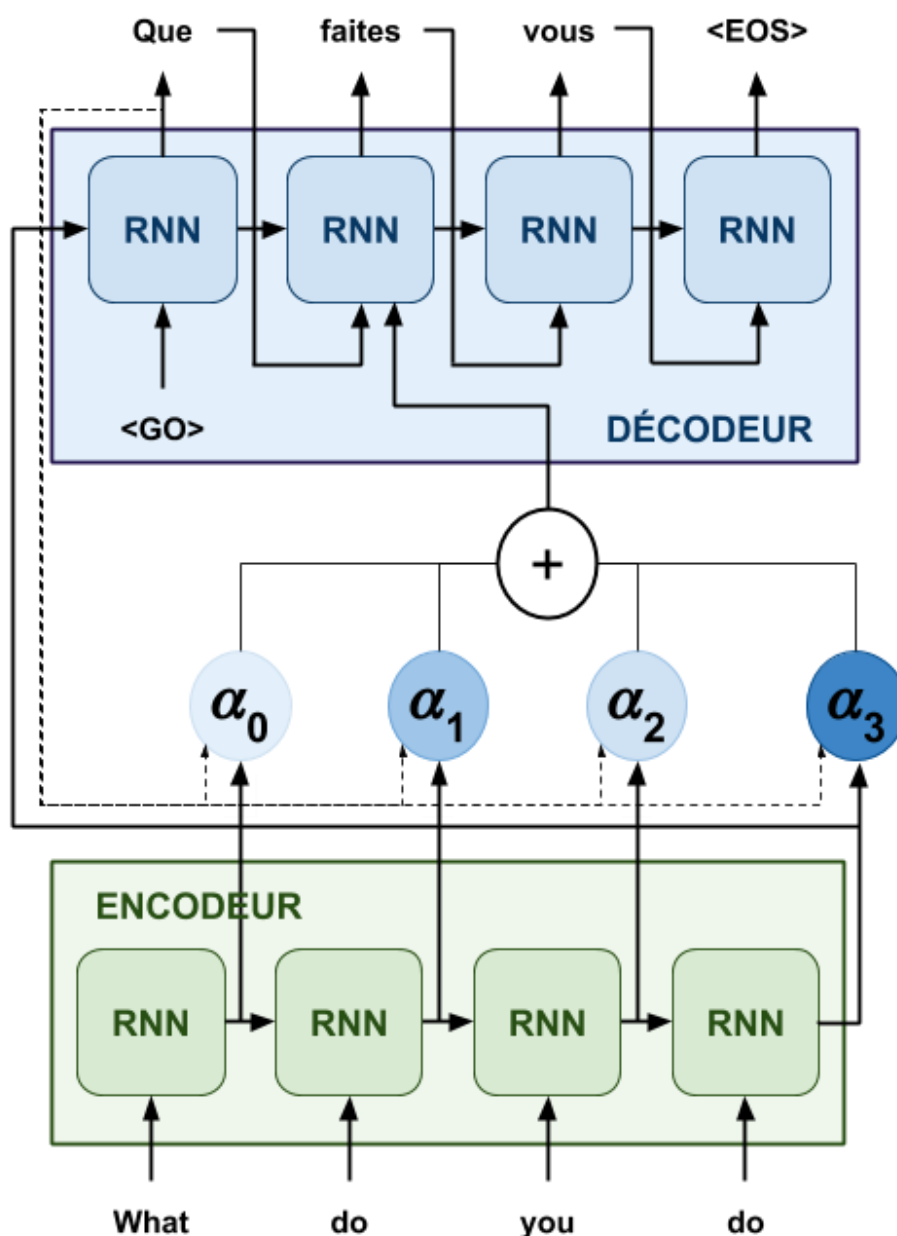


Figure 3.6 : Schéma du mécanisme d'attention, à l'étape où le décodeur génère le mot "faites".



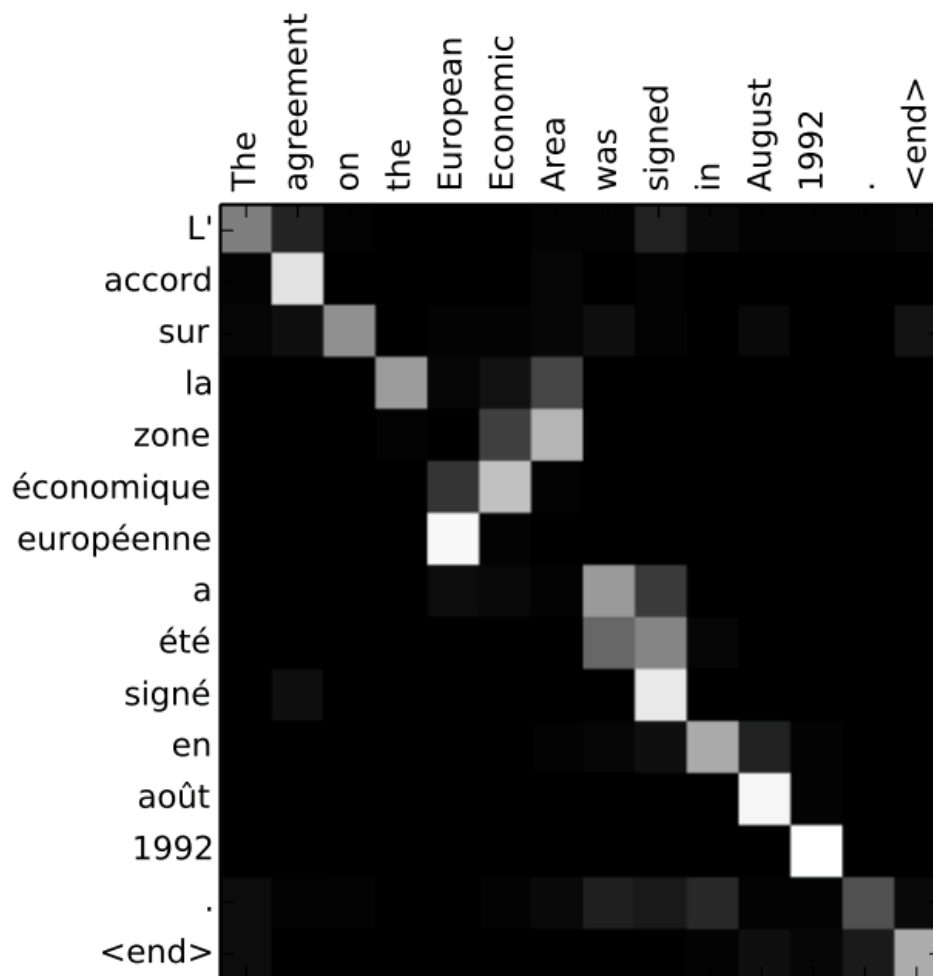


Figure 3.7 : Visualisation des scores d'attention calculés par un modèle de traduction automatique de l'anglais au français. Plus les carrés sont clairs, plus le score est élevé.

### 3.3.2 Architecture du Transformer

Comme dit plus haut, le Transformer est un modèle de type encodeur-décodeur basé entièrement sur le mécanisme d'attention. Comme un modèle encodeur-décodeur classique, le Transformer encode d'abord la séquence d'entrée en une représentation numérique, pour ensuite la décoder en une autre séquence de *tokens*. La différence avec les modèles étudiés dans la Partie 3.2 est que la récurrence n'est pas utilisée. Autrement dit, le modèle traite la séquence d'entrée d'un seul coup, plutôt que de la traiter *token* par *token*. C'est donc uniquement grâce à l'utilisation du mécanisme d'attention que le Transformer peut réussir à comprendre des dépendances avec des *tokens* passés.

La Figure 3.8 nous présente plus en détail l'architecture du Transformer. On voit dans la partie a) de la figure que les parties encodeur et décodeur du modèle sont toutes deux composées d'une pile de blocs encodeurs et décodeurs, respectivement. Comme dans un encodeur-décodeur récurrent, la phrase d'entrée est d'abord traitée par les encodeurs. L'information résultant du traitement du dernier encodeur est ensuite envoyée aux blocs décodeurs pour générer la phrase. On voit dans la partie b) de la Figure 3.8 le détail des blocs encodeurs et décodeurs. Un bloc encodeur est composé d'abord d'une

couche de “**self-attention**” (ou “auto-attention” en français). Celle-ci permet à l’encodeur de se concentrer sur certains mots de la phrase d’entrée pendant l’encodage d’autres mots. Prenons comme exemple la phrase “l’homme n’est pas rentré dans le magasin car il n’avait pas de masque”. À quoi correspond le mot “il” dans cette phrase : à “l’homme”, au “magasin” ou au “masque”. Pour un être humain parlant français, la réponse paraît évidente, mais pas pour l’algorithme. Grâce à la couche de *self-attention*, le bloc encodeur peut donc se concentrer sur les *tokens* “l” et “homme” pour encoder le mot “il”. L’encodeur regarde donc tous les *tokens* pour créer un meilleur encodage de chaque mot. Le résultat est ensuite passé dans un **réseau de neurones** (ou “*feed forward neural network*” en anglais) pour créer l’*output* envoyé au prochain bloc encodeur.

Le bloc décodeur est constitué des mêmes blocs que l’encodeur, mais il a en plus, entre les deux blocs, une couche d’**attention encodeur-décodeur**. Celle-ci permet au décodeur de se concentrer sur certaines parties de l’*input* pour générer chaque mot de la séquence de sortie, à la manière de l’encodeur-décodeur avec attention présenté dans la Partie 3.3.1. La pile de décodeur prend entrée chaque mots générés pour prédire le prochain, à la manière d’un décodeur récurrent.

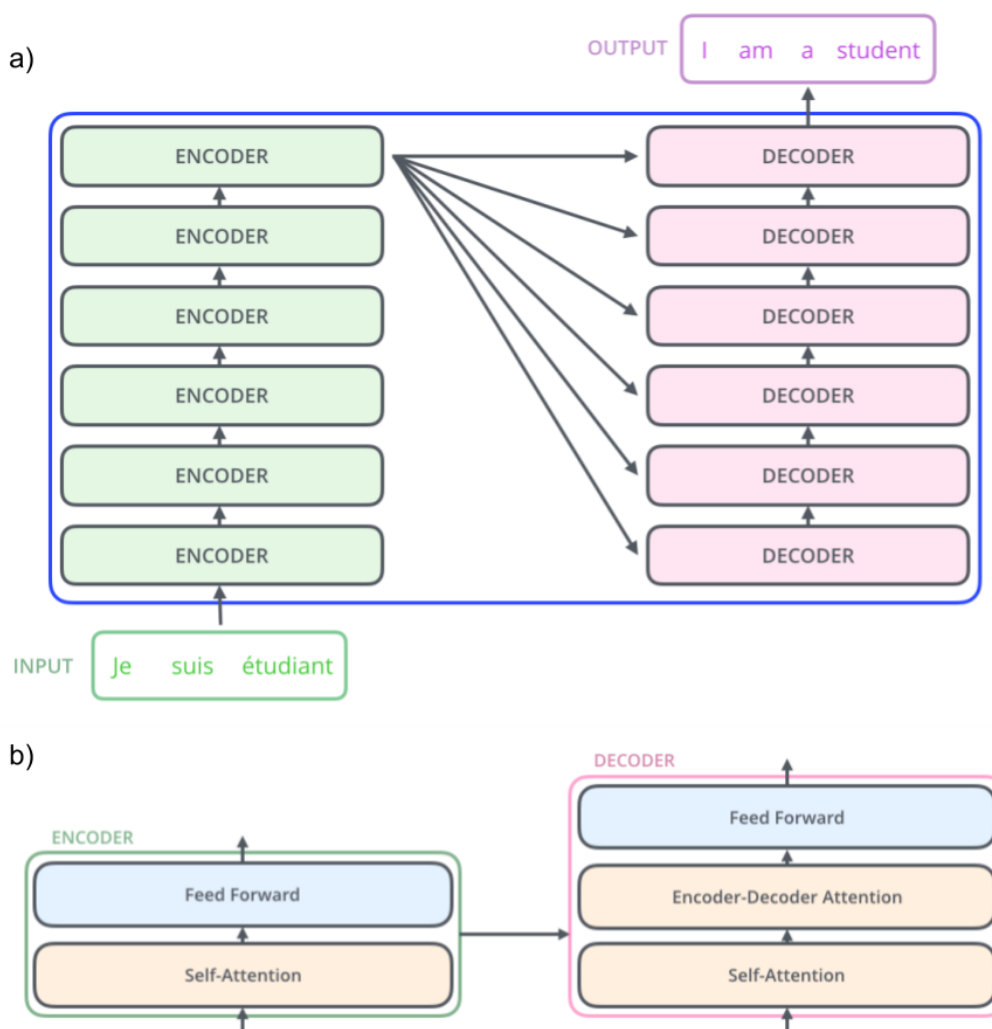


Figure 3.8 : a) Schéma global d’un Transformer avec ses piles de blocs encodeurs et décodeurs. b) Schéma des différents composants des blocs encodeur et décodeur du Transformer. [4]

Toutes les techniques décrites dans cette partie font partie des fondements des méthodes à la pointe de l'état de l'art en NLG. Comme on le verra plus bas, il y a de nombreuses manières d'appréhender les différents problèmes de NLG. Il est donc important de bien comprendre les techniques utilisées pour saisir les raisons motivant les différentes approches.

## 4. État de l'art

Les techniques neuronales présentées dans la partie précédente offrent un éventail très large de possibilités pour traiter les différentes tâches de la NLG. Dans les approches à l'état de l'art, on trouve de nombreuses méthodes différentes, donnant toutes des résultats concluants. Celles-ci se démarquent par différents moyens : leurs résultats meilleurs que les autres, leur capacité à répondre à de multiples tâches de la NLG, ou encore leur approche particulière d'un problème précis. Toutes sont intéressantes, soit parce qu'elles nous montrent le meilleur moyen connu à ce jour pour résoudre une tâche en particulier, soit parce qu'elles donnent un aperçu des différentes directions qui seront prises dans le futur. Dans cette partie, nous allons donc nous pencher sur les différentes méthodes utilisées dans la recherche récente pour résoudre les nombreux problèmes de NLG. Nous nous concentrerons donc sur les méthodes neuronales, c'est-à-dire celles qui utilisent des réseaux de neurones, car ceux sont celles qui donnent les meilleurs résultats. D'autre part, le but final de ma mission étant de réaliser un système conversationnel, nous étudierons en majeure partie des publications scientifiques dédiées à cette tâche. De plus, nous nous concentrerons sur l'étude de modèle traitant de l'anglais car c'est la langue qui voit le plus de développements en NLP et qui compte la plus grande quantité de données. Il est important de noter que cet état de l'art ne se veut en aucun cas exhaustif. Le but est avant tout d'explorer différentes techniques pour résoudre les tâches qui nous intéressent.

### 4.1 *Transfer Learning*

Traditionnellement, les paramètres des modèles de DL sont initialisés aléatoirement. Le *transfer learning* (ou « apprentissage par transfert » en français) est une méthode du ML qui consiste à utiliser un modèle **pré-entraîné** sur une tâche, pour réaliser une nouvelle tâche. L'idée est de récupérer une partie des paramètres appris par le modèle sur la première tâche pour apprendre la seconde plus rapidement, ce qui nécessite donc moins de données. Le *transfer learning* a montré sa capacité à améliorer grandement les performances de certains modèles en tirant partie de cette phase de pré-entraînement. En NLP, le *transfer learning* sert à donner au modèle une compréhension du langage avant de le transférer sur une tâche plus précise. Une phase de pré-entraînement est donc réalisée durant laquelle le modèle s'entraîne sur d'énormes quantités de texte brut annoté automatiquement (voir Note 4.1). Une fois que le modèle est pré-entraîné, son architecture est légèrement modifiée, si nécessaire, pour réaliser la ou les tâches voulues. L'avantage de cette méthode se trouve surtout dans les données utilisées pendant l'entraînement des modèles. En effet, en NLP, les corpus annotés sont très coûteux à réaliser et sont restreints en quantité de données. En revanche, il est extrêmement facile de trouver de très grandes quantités de texte brut : l'internet en est une source quasi-infinie. Il est donc judicieux de construire d'abord un modèle de langue sur ces grandes quantités de données, pour ensuite affiner les paramètres du modèle sur une tâche plus spécifique. On parle alors d'entraînement « **semi-supervisé** » (voir Note

4.1), avec une phase de pré-entraînement sur un gros corpus brut annoté automatiquement, suivie par un entraînement sur un corpus plus restreint annoté manuellement. Cette technique donne les modèles les plus performants de l'état de l'art en NLP. On peut citer notamment BERT, pour « *Bidirectional Encoder Representations from Transformers* », (Devlin et al., 2018) qui utilise la partie encodeur du Transformer de Vaswani et al. (2017) et est à la base de certaines des meilleures méthodes de l'état de l'art en NLP.

#### Note 4.1 : supervision et annotations

En ML, il est possible de faire deux sortes d'apprentissage : supervisé et non-supervisé. En apprentissage **supervisé**, l'objectif est d'apprendre une fonction qui, compte tenu d'un échantillon de données et des résultats souhaités, se rapproche le plus possible de la relation entre l'entrée et la sortie observable dans les données. On dit que les données sont **annotées**, les annotations étant les *outputs* que notre modèle doit produire en fonction des *inputs* correspondantes. En apprentissage **non-supervisé**, les données ne sont pas annotées. Le but du modèle est donc de déduire la structure naturelle présente dans un ensemble de données.

La tâche de pré-entraînement consiste à modéliser la langue. Comme nous l'avons dit dans la Partie 3, cela consiste à associer à des mots, expressions ou phrases, une probabilité, selon une étude statistique de corpus. Cette tâche a l'énorme avantage qu'elle peut être réalisée de manière « **auto-supervisée** » : dans un corpus de texte, chaque mot, expression ou phrase est sa propre annotation. Il est donc possible d'utiliser cette propriété pour annoter automatiquement de très larges corpus de texte et donc apprendre un modèle de langue de manière complètement automatisée. La méthode du « modèle de langue masqué » (MLM) est par exemple beaucoup utilisée (Devlin et al., 2018 ; Raffel et al. 2019 ; Song et al., 2019 ; Zhang et al., 2019) : dans chaque séquence du corpus de pré-entraînement, des mots sont choisis au hasard pour être masqués. Le modèle devra tenter de prédire ces mots masqués. Cette méthode, ainsi que d'autres plus ou moins efficaces (Howard and Ruder, 2018 ; Devlin et al., 2018 ; Radford et al., 2018 ; Wolf et al., 2019) permettent de construire un modèle de langue de manière totalement automatisée. Une fois que cela est fait, le modèle peut être affiné sur la tâche désirée.

Le but de cette approche de pré-entraînement est de pouvoir réutiliser une architecture très facilement pour un grand nombre de tâches différentes. En témoignent les deux benchmarks GLUE (Wang et al., 2018) et SuperGLUE (Wang et al., 2019) qui cherchent à évaluer la compréhension du langage naturel des modèles en rassemblant chacun une dizaine de tâches comprenant notamment de l'analyse de sentiments, l'implication textuelle ou la synthèse de texte. Les modèles évalués par ces benchmarks, qu'on peut retrouver sur le site de GLUE et SuperGLUE [5], démontrent la capacité des modèles pré-entraînés à s'adapter facilement à de nombreuses tâches.

Au sein de tous les modèles exploitant le *transfer learning*, certains mettent la NLG au centre de leur approche. Raffel et al. (2019) proposent leur « *Text-To-Text Transfer Transformer* » (T5) qui unifie un grand nombre de tâches de NLP dans une approche *text-to-text*. Les tâches sont toutes transformées pour que leur input et leur output soient sous forme de texte (voir Figure 4.1) : l'input est mis sous la forme « <tâche> : <texte d'input> » et l'output est traduit en texte pour les tâches non génératives (classification ou régression). Leur modèle est un Transformer Encodeur-Décodeur classique pré-entraîné avec une version optimisée de la méthode MLM. Le corpus de pré-entraînement

a été créé pour ce papier : le *Colossal Clean Crawled Corpus* (C4) rassemble 750 GB de texte provenant d'internet, méticuleusement nettoyé par les auteurs. Finalement l'approche porte ses fruits puisqu'en plus d'être très facilement utilisable pour de nombreuses tâches, elle donne de très bons résultats dans chacune d'entre elles. En effet, à sa publication, T5 s'est positionné en première place des benchmarks GLUE et SuperGLUE (à l'heure de l'écriture de cette étude, ils ne sont plus que deuxième de GLUE).

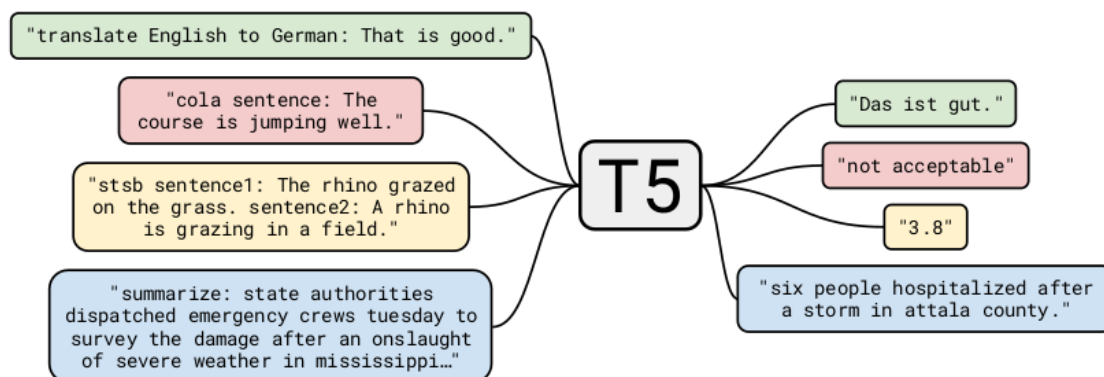


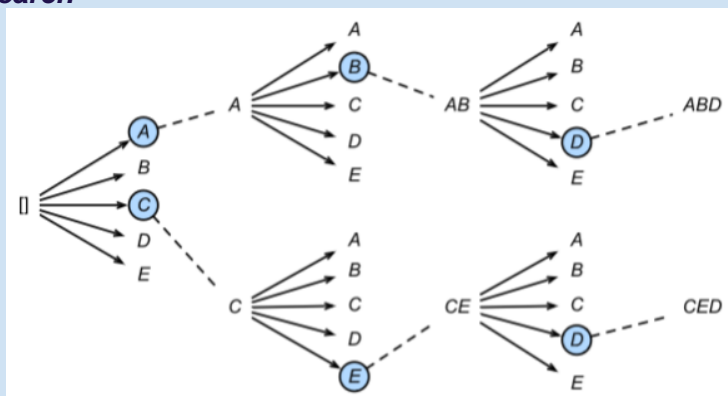
Figure 4.1 : Illustration de l'unification des tâches sous une forme *text-to-text* de l'architecture T5 de Raffel et al. (2019).

L'approche de Zhang et al. (2019) se focalise sur la synthèse de texte abstractive. Cette tâche consiste à générer un résumé d'un ou plusieurs textes, en essayant de reformuler au maximum leur contenu. Pour ce faire, les auteurs proposent leur modèle PEGASUS. Celui-ci utilise également l'architecture du Transformer Encodeur-Décodeur. Pour la phase de pré-entraînement, en plus de la méthode MLM, les auteurs entraînent le modèle à générer des phrases entièrement masquées dans le texte d'origine. Les auteurs montrent que le fait de choisir les phrases masquées selon leur importance dans le texte a un impact positif sur la qualité de la synthèse. Autrement dit, ils s'aperçoivent que réaliser une tâche de pré-entraînement proche de la tâche finale donne de meilleurs résultats. Sur la tâche de synthèse de texte, PEGASUS surpasse toutes les autres approches avec les meilleurs scores de l'état de l'art.

Des modèles appliquent le pré-entraînement d'un Transformer aux agents conversationnels. TransferTransfo (Wolf et al., 2019) et DialoGPT (Zhang et al., 2019b) modèles récupèrent le Transformer pré-entraîné par Radford et al. (2018) et le ré-entraînent pour la tâche de conversation. Wolf et al. (2019) utilisent pour cela le data-set Persona-Chat (Zhang et al., 2018a). Ce dernier rassemble des dialogues entre personnes conditionnées par un profil. Pour générer une réponse à la discussion en cours, TransferTransfo concatène les phrases du profil de l'agent et les cinq dernières déclarations du dialogue. Avec cet input, le modèle peut générer une réponse en accord avec le profil correspondant et avec le contexte de la discussion. Pendant l'entraînement, un classifieur est entraîné à distinguer, entre quelques réponses choisies au hasard, celle qui correspond au dialogue en cours. Ce classifieur est ensuite utilisé pendant la phase de test pour classer les différents *beams* d'un *beam search* (voir Note 4.2). Dans DialoGPT, Zhang et al. (2019b) intègrent un modèle "**backward**" ("inverse" en français) qui génère la requête à partir de la réponse générée. Pendant la génération, plusieurs réponses possibles sont générées, le modèle *backward* tente de générer la requête en

fonction de toutes ces réponses. Celle qui donne le résultat le plus proche de la vraie requête est choisie. Cela permet d'assurer la cohérence des réponses avec leur requête, mais aussi de générer des réponses moins génériques. En effet, les phrases très génériques comme "je ne sais pas" sont associées avec beaucoup de requêtes différentes. Elles donneront donc de mauvais résultats avec le modèle *backward*.

#### Note 4.2 : *beam search*



Un **beam search** est une technique de décodage pour générer une phrase. Elle utilise la sortie du modèle pour générer plusieurs options de phrases à générer, appelées **beams**. À chaque étape de la génération, l'algorithme choisit plusieurs options possibles pour continuer chaque *beam* (voir image ci-dessus). On associe ensuite un score à chaque *beam* qui correspond à sa qualité par rapport à la tâche. Le *beam* avec le meilleur score est choisi pour être généré.

Roller et al. (2020) proposent également leur chat-bot *Blender* qui tire parti du *transfer learning*. D'abord, leur modèle, un Transformer classique, est pré-entraîné sur un large corpus de discussions de groupe. Ensuite, les auteurs montrent que la qualité des données utilisées durant la phase d'affinage du modèle sur la tâche finale est cruciale. En s'entraînant sur des données qui mettent en avant des compétences importantes à avoir dans une conversation, comme l'empathie, la cohérence de la personnalité, le fait d'écouter son partenaire ou de montrer des connaissances ; les modèles montrent de bien meilleurs résultats. *Blender* est donc entraîné sur plusieurs data-sets dans le but d'apprendre les compétences liées à différentes tâches :

- Persona-Chat (Zhang et al., 2018a), concentré sur la cohérence de la personnalité,
- *Empathetic Dialog* (Rashkin et al., 2019), concentré sur l'empathie,
- *Wizard of Wikipedia* (Dinan et al., 2018), concentré sur l'apport de connaissances factuelles au modèle,
- *Blended Skill Talk* (Smith et al., 2020) qui mélange toutes les tâches précédentes.

Les auteurs montrent également que la longueur des déclarations d'un agent conversationnel a une grande influence sur les évaluations humaines de la qualité du modèle. En contraignant celui-ci à générer des déclarations plus longues, les phrases générées sont moins génériques et plus intéressantes et captivantes. Finalement, *Blender* est sans doute le meilleur système conversationnel existant. Il existe en trois versions différant sur le nombre de paramètres du modèle : 90 millions, 2,7 milliards ou 9,4 milliards. Nous avons essayé la plus petite version et elle montrait déjà des résultats impressionnants, bien meilleurs que les autres modèles que nous avons essayés.

## 4.2 Apport de connaissances au modèle

Jusqu'ici, les modèles que nous avons étudiés ne se basaient que sur leur modèle de langue et le texte en input pour générer du texte en sortie. Or, que ce soit pour pallier un problème ou répondre à une tâche bien précise, on peut vouloir apporter des connaissances extérieures au modèle pour influencer sa génération. Il faut donc répondre à plusieurs questions : Quelles informations extérieures apporter à notre modèle ? Comment encoder ces informations pour qu'elles soient utilisables par le modèle ? Et comment les utiliser pour influencer la génération ? En fonction du problème de base, différentes solutions seront envisagées.

Comme nous l'avons vu, les approches neuronales ont l'énorme avantage de permettre la construction d'un modèle de langue et la génération de texte de manière complètement automatisée, et ce en ne se basant que sur l'étude de corpus de texte. Néanmoins, le fait de se baser sur cette étude statistique de corpus entraîne un problème récurrent : les textes générés sont souvent très génériques ou pauvres sémantiquement. En effet, une grande partie des corpus de texte correspond à des phrases ou expressions types qui ne portent pas de sens à elles-seules. Ce problème est particulièrement présent dans les systèmes conversationnels. On trouvera très souvent dans les corpus des réponses comme « je ne sais pas » ou « moi aussi » (Li et al., 2015). Ces réponses se retrouvent souvent dans les sorties des modèles les plus simples, à cause de leur très grande fréquence dans les corpus de texte. Pour répondre à ce problème, beaucoup de solutions sont envisagées. L'apport de connaissances au modèle est notamment une approche prometteuse. En le forçant à mentionner des informations ou thèmes spécialement choisis en fonction de la discussion en cours, le modèle produira des phrases plus intéressantes et captivantes.

### 4.2.1 Contexte

Il est possible d'apporter différents types de connaissances. Une première approche très développée cherche à introduire dans le modèle une conscience du contexte de la tâche. Dans le cadre d'un système conversationnel, le contexte correspond aux informations évoquées plus tôt dans la conversation. Les systèmes les plus classiques de type Encodeur-Décodeur génèrent une réponse à partir de la dernière déclaration. Ces modèles n'ont donc accès qu'à une fraction du contexte parfois très petite, voire nulle. Des approches se développent donc pour pouvoir à la fois extraire le contexte d'une conversation et amener des informations supplémentaires lors de la génération. Tian et al. (2017) distinguent ces approches en deux catégories (voir Figure 4.2). Une première, dite "non-hiérarchique", qui concatène simplement l'ensemble des séquences passées de la conversation pour construire l'input du modèle. Une deuxième, dite "**hiérarchique**", nettement plus efficace, qui décompose l'input du générateur en plusieurs. Un premier encodeur s'occupe d'encoder la dernière déclaration du dialogue, comme dans une architecture Encodeur-Décodeur classique. Un deuxième modèle encode le contexte de la conversation en se basant sur les outputs du premier encodeur. Plusieurs publications implémentent ces modèles hiérarchiques avec des RNN simples (Serban et al., 2016) ou des LSTM (Dušek and Jurčiček, 2016). Tian et al. (2017) présentent également leur modèle hiérarchique qui pondère l'importance des encodages de la dernière séquence et du contexte avec un score d'attention. Leur but est de se concentrer sur la dernière séquence si le contexte n'est pas important, et inversement. Mei et al. (2017) proposent une autre approche pour intégrer le contexte à la génération.



D'abord, le modèle n'est plus un Encodeur-Décodeur mais un modèle de langue de type RNN : il prédit le prochain mot, après avoir traité tous les *tokens* depuis le début de la conversation. Selon les auteurs, un modèle de langue permettrait donc de mieux comprendre comment une conversation évolue sur toute sa durée. A ce RNN est ajouté un mécanisme d'attention sur tous les *tokens* passés, pour donner plus d'importance à certains mots utilisés depuis le début de la conversation. Enfin, toujours pour valoriser le contexte du dialogue, les auteurs extraient les sujets évoqués dans la conversation avec l'algorithme *Latent Dirichlet Allocation* (LDA, ou "allocation de Dirichlet latente" en français). Les sujets trouvés servent à classer les prédictions du modèle.

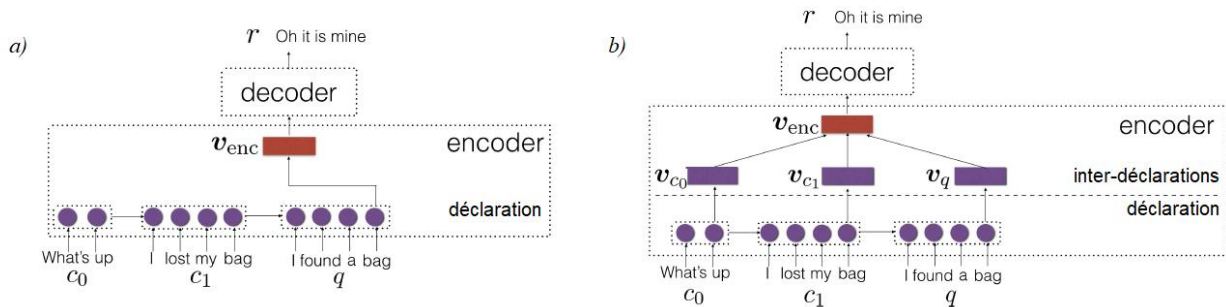


Figure 4.2 : a) Modèle non-hiérarchique, avec un encodeur classique qui encode chaque *token* successivement, l'output final de la requête ( $q$ ) sera l'input du générateur. b) Modèle hiérarchique, avec un modèle qui encode chaque déclaration et un deuxième modèle qui encode le contexte en se basant sur les outputs du premier encodeur. Le vecteur en résultant sera l'input du générateur. (Tian et al., 2017)

#### 4.2.2 Informations factuelles et mémoire

En plus du contexte, il est également possible de chercher à apporter des connaissances extérieures au modèle, par exemple sous la forme d'informations factuelles. Dans cette optique, Xing et al. (2017) utilisent également l'algorithme LDA pour créer un modèle de conversation de type Encodeur-Décodeur "conscient du sujet" (*topic-aware* en anglais). Avec une étude d'un corpus, les auteurs associent à chaque sujet des mots en fonction de leur nombre d'apparition dans le sujet en question. Dans le cadre d'un dialogue, le sujet principal de la requête est extrait et les mots correspondants sont utilisés pour conditionner le décodeur avec un mécanisme d'attention. D'autres publications utilisent des approches similaires, comme Ghazvininejad et al. (2018) qui utilisent une base d'avis de clients sur des restaurants pour améliorer les recommandations d'un chat-bot. De la même manière, plusieurs papiers (Dinan et al., 2018 ; Parthasarathi and Pineau, 2018) récupèrent de l'information dans des articles Wikipedia liés au contexte de la conversation pour rendre les réponses de leur modèle plus informatives. D'autres approches cherchent à rendre leur système de dialogue moins générique en utilisant une base de connaissances relevant du sens commun (Young et al., 2018 ; Zhou et al., 2018). Des informations comme "Paris est la capitale de la France" ou "un chien est un animal de compagnie" sont traduites sous la forme de graphe dans des bases comme ConceptNet (Speer et al., 2017). En reliant ces informations au contexte de la conversation, le modèle peut choisir des réponses plus adaptées et moins génériques.

L'utilisation de bases de connaissances extérieures au modèle est particulièrement développée dans les **Memory Networks** (MN, ou "réseau à mémoire" en français). Définis par Weston et al. (2014), les MN sont des réseaux utilisant une base de connaissances dynamique : celle-ci est modifiée en fonction des inputs du modèle et



utilisée pour conditionner la génération de texte, permettant une meilleure généralisation face à de nouveaux cas d'utilisation. Comme nous allons le voir, cette définition très générale permet une grande diversité dans les techniques utilisées. He et al. (2017) exploitent les graphes dans le cadre de la génération d'un dialogue. Les deux agents conversant ont accès à une petite base de connaissances sous la forme d'une liste de relations entre des entités. Leur but est de déterminer laquelle de ces relations ils ont en commun. Les auteurs associent à chaque partie du graphe un score d'attention qui conditionnera le modèle Encodeur-Décodeur lors de la génération. Ce score d'attention est dynamique : au cours de la conversation, les différentes entités évoquées vont influencer sur leur score d'attention et celui de leurs relations dans le graphe. L'idée est d'inciter le modèle à reconnaître des entités connues et de les relier aux autres entités présentes dans sa base de connaissances. Plusieurs autres approches utilisent des techniques de *clustering* pour construire une mémoire. Tian et al. (2019) encodent les couples <requête, réponse> des conversations et déterminent des *clusters* (groupes) de séquences avec l'algorithme *K-Means*. Le but est de déterminer des groupes de séquences similaires et d'utiliser l'encodage des réponses correspondantes pour conditionner le décodeur. Khandelwal et al. (2020) utilisent l'algorithme de *clustering K Nearest Neighbors* (KNN) pour conditionner un modèle de langue déjà entraîné. Le modèle prédit le prochain mot d'une séquence en se conditionnant sur des contextes similaires vus pendant l'entraînement du modèle KNN. L'avantage de cette approche est qu'elle rajoute une couche à un modèle de langue sans le ré-entraîner. On peut donc par exemple décider d'adapter un modèle à une tâche très précise en ajoutant la couche KNN entraînée sur un corpus de texte correspondant à la tâche en question.

Les déclarations dans une conversation seront toujours très liées au contexte du dialogue et aux dernières déclarations. Le caractère dynamique des MN permet donc d'utiliser à la fois les informations précédemment évoquées et des connaissances extérieures au modèle. De plus, les MN sont très flexibles, que ce soit dans leurs architectures que dans les données qu'ils peuvent apporter aux modèles. Cela permet d'abord une grande liberté dans les techniques utilisées. Celles-ci sont donc parfois largement maîtrisées par la communauté scientifique, comme on l'a vu avec les méthodes de *clustering*, ce qui permet une grande liberté dans l'innovation. De plus, les MN ne sont théoriquement pas limités dans les données qu'ils peuvent apporter au modèle : pas besoin d'avoir un corpus d'entraînement bien précis, de nouvelles données récupérées à n'importe quel moment après l'entraînement pourraient être utilisées pour rendre les réponses générées plus adéquates. En résumé, les MN semblent aujourd'hui très prometteurs, voire indispensables, dans la construction de systèmes conversationnels.

#### 4.2.3 Informations personnelles

Un problème récurrent des systèmes de NLG, et plus particulièrement des systèmes conversationnels, est l'inconsistance des personnalités au cours de la génération. Pendant une conversation, les participants partagent des traits qui leur sont propres et qui sont constants : un lieu d'origine, des centres d'intérêts ou de simples opinions. Un chat-bot classique qui ne génère une réponse qu'à partir de la dernière déclaration sera incapable de conserver cette consistance dans sa personnalité. On verra alors des incohérences entre les déclarations avec par exemple "J'ai grandi à Paris" et plus tard "Je ne suis jamais allé en France". Pour résoudre ce problème, une solution est

d'apporter des informations au modèle sur la personnalité de l'agent. En conditionnant le modèle à générer du texte en rapport avec un profil, celui-ci sera plus consistant tout au long de la génération.

Pour inclure les informations d'un profil de l'agent conversationnel, une première approche est de réaliser un *embedding* des personnes pouvant prendre part aux conversations. Li et al. (2016a) créent un *embedding* de chaque locuteur d'un corpus à partir de caractéristiques personnelles (âge, sexe, dialecte, etc.). Le but est de relier ces caractéristiques à des mots et expressions souvent utilisés. Ils utilisent ensuite cet *embedding* pour conditionner un modèle de type Encodeur-Décodeur en fonction du locuteur et de l'interlocuteur. Kottur et al. (2017) récupèrent cette méthode dans un modèle hiérarchique (Serban et al., 2016 ; voir Partie 4.2.1) qui inclut donc également le contexte à la génération. Zheng et al. (2019) proposent deux approches pour améliorer cette méthode. La première introduit un mécanisme d'attention sur les différentes caractéristiques du profil du locuteur. La seconde introduit un biais sur l'output du décodeur qui utilise l'*embedding* du profil pour décider du niveau d'importance de celui-ci dans le prochain mot généré. Il peut donc décider de générer un mot en rapport avec un trait ou simplement un mot sémantiquement logique, donnant donc des phrases mieux formées.

Zhang et al. (2018a) explorent différentes méthodes en se basant sur des informations légèrement différentes. Leur data-set PERSONA-CHAT définit un profil pour chaque locuteur par quelques phrases caractérisant des goûts, des préférences ou des traits de personnalité. On ne dispose donc pas des mêmes informations pour tous les locuteurs, mais de phrases pouvant évoquer des sujets très divers (famille, cinéma, origine, etc.). Les auteurs proposent donc une méthode qui encode les phrases du profil avec les dernières déclarations pour les considérer comme du contexte. Cette méthode sera reprise avec une architecture de type Transformer dans le modèle TransferTransfo (Wolf et al., 2019) étudié dans la Partie 4.1.

#### 4.2.4 Informations affectives

Une partie importante du langage naturel est l'expression de nos sentiments et émotions. Un texte qui en est dépourvu pourra donc paraître très générique ou peu captivant. Plusieurs papiers cherchent donc à résoudre ce problème. Encore une fois, le problème se résume à déterminer quelles informations incorporer au modèle, et comment les incorporer. Ghosh et al. (2017) extraient la catégorie affective des mots d'après leur contexte avec le programme d'extraction de caractéristiques "*Linguistic Inquiry and Word Count*" (ou "enquête linguistique et compte de mots" en français). A partir de ces catégories, ils peuvent apprendre un *embedding* affectif de chaque mot qu'ils peuvent utiliser pour conditionner un modèle de langue sur une catégorie affective particulière. Zhou et al. (2017) donnent en input de leur modèle un vecteur d'émotion. Pour cela, un *embedding* de chaque émotion est appris pendant l'entraînement. Pendant la génération, un module de mémoire interne est utilisé pour traquer la quantité d'émotion exprimée dans les mots générés. De ce fait, le modèle s'assure que l'émotion demandée est bien transmise. Pour garder une cohérence sémantique dans la phrase, un module de mémoire externe est également utilisé pour conditionner le modèle à générer soit un mot porteur d'émotion, soit un mot générique pendant le décodage.

Plusieurs travaux basent leur approche sur la métrique **VAD** ("*Valence Arousal Dominance*") qui traduit la valeur affective des mots en trois valeurs réelles. Cette

métrique est particulièrement pratique pour les modèles de NLG, d'une part parce qu'elle est sous la forme d'un vecteur de nombres réels, donc très aisément incorporable dans un modèle de *deep learning*. Et d'autre part car il existe des dictionnaires de vocabulaire affectif recensant le VAD de dizaines de milliers de mots, valables dans toutes les langues. Asghar et al. (2017) utilisent le VAD sous trois formes pour créer un modèle conditionné par des émotions. D'abord, les trois valeurs du VAD de chaque mot sont concaténées à son *embedding*. Ensuite, les auteurs définissent une fonction de coût qui cherche à maximiser le contenu affectif dans le texte généré. Enfin, un *beam search* est effectué en se basant sur un score découlant du VAD pour s'assurer que le texte généré comporte le maximum de contenu émotionnelle. Zhong et al. (2019) récupèrent ces méthodes et ajoutent un mécanisme d'attention sur les mots affectifs de l'input du modèle, encore une fois en se basant sur le VAD.

### 4.3 Apprentissage par renforcement

En NLG, et particulièrement dans les systèmes conversationnels, l'objectif d'optimisation utilisé le plus souvent pour entraîner les modèles est la **Maximum Likelihood Estimation** (MLE, ou "estimation du maximum de vraisemblance" en français). Il correspond à chercher les paramètres du modèle qui associeront une probabilité maximale aux éléments du corpus d'entraînement. Autrement dit, le modèle est entraîné à générer des phrases similaires à celles vues dans le corpus d'entraînement. Plusieurs problèmes émergent de cet objectif d'optimisation. Nous avons déjà cité le premier dans la Partie 4.2 : la présence majoritaire de phrases génériques comme "je ne sais pas" ou "je ne sais pas de quoi tu parles" dans les corpus d'entraînement pousse les modèles à générer ces mêmes phrases trop souvent (Li et al., 2015). De plus, la MLE ne prenant pas en compte la répétition de phrases pendant une conversation, les systèmes conversationnels classiques se retrouvent souvent dans des boucles de répétition des mêmes déclarations. Enfin, les algorithmes classiques de NLG souffrent d'une incohérence profonde due au fait que l'objectif d'optimisation pendant l'entraînement n'est pas le même que les mesures de qualité du modèle pendant la phase de test. Il est donc souvent difficile de faire converger les modèles vers un objectif qui n'est pas traité pendant l'entraînement.

Pour répondre à ces problèmes, il faut trouver des alternatives aux modèles basés sur l'optimisation de la MLE. Une de ces alternatives est le **Reinforcement Learning** (RL, ou "apprentissage par renforcement" en français). Le RL est un champ de recherche du ML qui apprend une stratégie à un agent par le biais d'expériences répétées. Pour chaque action prise, l'agent reçoit une récompense positive ou négative. Son but est de maximiser ses récompenses à long terme. Le RL a de nombreux avantages sur les théories d'apprentissage supervisé et non-supervisé, plus largement employées. Il permet notamment d'optimiser des fonctions conçues spécialement pour la tâche en question grâce au système de récompenses. Contrairement aux algorithmes de ML plus classiques, les méthodes du RL permettent d'optimiser des fonctions non-différentiables, par le biais d'**estimations des récompenses** sur le long terme. Cela procure aux développeurs une plus grande liberté dans la création des modèles et de leurs objectifs d'optimisation. Cela permet aussi d'introduire des métriques de test comme objectif pour les modèles.

Plusieurs approches (Bahdanau et al., 2016 ; Li et al., 2016b ; Srinivasan et al., 2019) utilisent le RL pour améliorer les résultats d'un modèle déjà entraîné à la génération

de texte. Li et al. (2016b) entraînent un système conversationnel classique de type Encodeur-Décodeur avec attention, puis ajoutent la couche de RL en ré-entraînant le modèle pour maximiser trois récompenses : la facilité pour répondre au texte généré, qui vise à ne pas bloquer la conversation ; la quantité d'information dans le message et la cohérence sémantique du texte. Srinivasan et al. (2019) adoptent une méthode très similaire résumée dans la Figure 4.3. Ils échangent la quantité d'information par l'intelligence émotionnelle, qui vise à conserver une cohérence émotionnelle entre l'input et l'output du modèle ; et ajoutent une récompense externe basée sur des évaluations humaines qui mesure l'utilité de la réponse générée. Les deux modèles cherchent à maximiser ces récompenses (*rewards* en anglais) en répétant un grand nombre d'expériences.

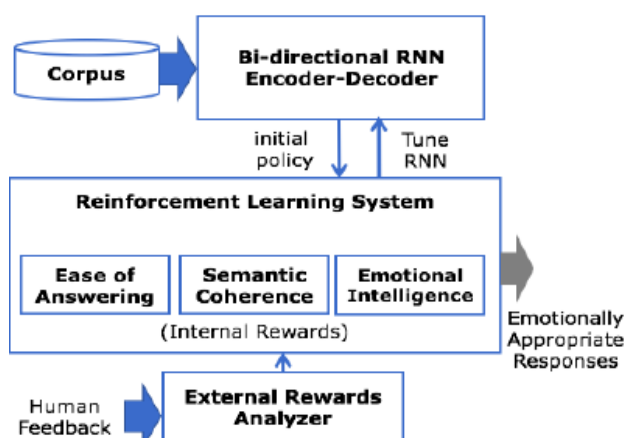


Figure 4.3 : Architecture de Srinivasan et al. (2019), leur Encodeur-Décodeur de type RNN Bi-directionnel détermine la stratégie initiale, le module de RL l'affine avec les trois récompenses internes : facilité de réponse, cohérence sémantique et intelligence émotionnelle ; ainsi que la récompense externe basée sur des évaluations humaines, jugeant l'utilité du message.

Bahdanau et al. (2016) adoptent une autre approche avec l'architecture de RL *Actor-Critic*. Celle-ci est composée de deux modèles : un Acteur qui choisit une action à réaliser et un Critique qui juge la qualité de l'action choisie, permettant donc à l'Acteur de s'améliorer au fil des expériences. Ici l'Acteur et le Critique sont deux modèles de type Encodeur-Décodeur. L'Acteur est entraîné à générer du texte dans le cadre d'une tâche de traduction automatique. Le Critique est entraîné à prédire le score BLEU du texte généré. Ce score BLEU est une métrique couramment utilisée pour évaluer les systèmes de traduction automatique. Le modèle optimise donc la même métrique qui sera utilisée pour évaluer le modèle. Les auteurs montrent que cela influe grandement sur les résultats, avec un meilleur score BLEU et beaucoup moins d'*overfitting*.

#### 4.4 Adversarial Learning

Toujours en réponse au problème des réponses génériques, une partie de la recherche en systèmes conversationnels s'est tournée vers l'*adversarial learning*. Le principe de cette approche, d'abord développée en *Computer Vision* avec les *Generative Adversarial Networks* (GAN, ou "réseaux adverses génératifs" en français), est d'utiliser deux modèles distincts (voir Figure 4.4). Le premier est le **générateur** : il s'occupe de générer l'objet souhaité (image en *Computer Vision*, texte en NLG). En NLG, ce

générateur est souvent similaire aux modèles utilisés dans les approches plus classiques : modèle de langue ou Encodeur-Décodeur. Le second modèle est le **discriminateur** : il classe l'objet généré, son rôle étant de distinguer au mieux les objets réels (générés par des humains) des objets générés par le générateur. Les deux modèles apprennent à réaliser leur tâche simultanément. De ce fait, chaque modèle devient progressivement meilleur à sa tâche, compliquant alors la tâche de l'autre, qui cherchera donc à s'améliorer, et ainsi de suite.

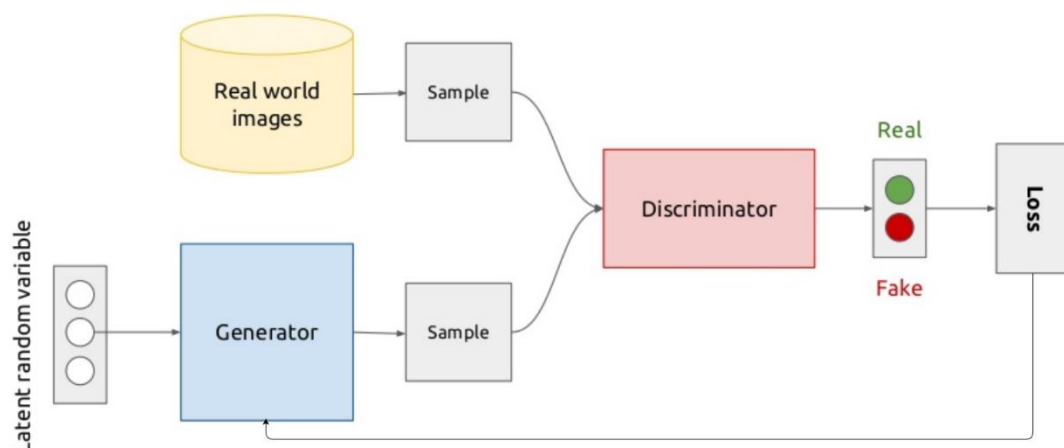


Figure 4.4 : Schéma d'un GAN, le générateur génère une image à partir d'un vecteur aléatoire, le discriminateur prend ces images ainsi que des images réelles et s'entraîne à les distinguer. L'output du discriminateur sert de fonction de coût (en anglais *loss*) pour le générateur. [6]

En NLG, le choix de l'*adversarial learning* paraît assez naturel : comme nous l'avons dit dans l'Introduction, le but ultime de tout système de génération de texte est d'être indifférenciable d'un être humain. Ici, le discriminateur prend un rôle de juge du texte généré, optimisant directement la différence entre les textes écrits par des humains et ceux générés artificiellement. Mais au-delà de cette intuition, l'*adversarial learning* répond également à une autre problématique évoquée en Partie 4.3 : trouver d'autres objectifs d'optimisation que la MLE, plus proches des métriques utilisées dans la phase de test des modèles. En essayant de réaliser au mieux une tâche s'apparentant au test de Turing (déterminer si notre interlocuteur est un humain ou un ordinateur), le discriminateur optimise une mesure qui s'approche des évaluations humaines des modèles de NLG. De plus, le RL, qui lui aussi cherchait à optimiser les métriques utilisées dans la phase de test, ne permet que de maximiser des récompenses définies manuellement par les développeurs. Li et al. (2017) arguent que cette approche ne permet pas de saisir tous les aspects de la NLG, notamment dans le cadre d'un dialogue. L'*adversarial learning* permettrait donc d'utiliser les outils du DL pour saisir ces aspects si cruciaux qui font qu'un texte semble être écrit par un être humain.

Une fois l'approche d'*adversarial learning* choisie, il faut trouver les méthodes pour la mettre en œuvre. Une première option est d'exploiter le RL pour utiliser l'output du discriminateur (Li et al., 2017 ; Zhang et al., 2018b ; Wu et al., 2019). Comme montré dans la Partie 4.3, le RL permet d'optimiser des objectifs externes au modèle par le biais de récompenses données à ce dernier pour chaque action choisie. Ici, au lieu de définir une fonction de récompense manuellement, la récompense est définie directement par l'output du discriminateur. Chez Li et al. (2017), le discriminateur prédit la probabilité que le texte en input soit écrit par un humain. Cette valeur sert donc de récompense pour le

générateur, qui cherchera à la maximiser pendant son entraînement. Zhang et al. (2018b) cherchent à générer du texte informatif en maximisant l'information mutuelle entre la source (texte en input) et le texte généré. Pour ce faire, le discriminateur encode la source, le texte généré et la cible (réponse réelle à la source), avec pour but de maximiser la différence entre la similarité [source-cible] et [source-texte généré]. L'objectif du générateur sera ensuite de minimiser cette même différence. Enfin, Wu et al. (2019) entraînent un discriminateur à déterminer si trois actes de dialogue (couples requête-réponse) sont ordonnés ou non. Le générateur reçoit comme récompense la probabilité que le dialogue qu'il a généré soit ordonné, et cherche à maximiser cette récompense. Les auteurs montrent que cette stratégie permet de mieux saisir la structure d'un dialogue à l'échelle de plusieurs actes de dialogues.

Une deuxième approche reprend l'objectif initial des GAN (Olabiya et al., 2019a ; Olabiya et al., 2019b) : une fonction de coût définit la capacité du discriminateur à différencier des textes réels et artificiels. Le discriminateur s'entraîne donc à maximiser cette fonction, tandis que le générateur s'entraîne à la minimiser. Olabiya et al. (2019a) adoptent cette méthode avec comme générateur un modèle hiérarchique (Serban et al., 2016 ; voir Partie 4.2.1). Pendant la phase de test, du bruit est ajouté à l'input pour générer plusieurs réponses. Celle qui obtiendra le meilleur score selon le discriminateur est choisie. Olabiya et al. (2019b) adoptent la même architecture en ajoutant en input du générateur et du discriminateur des informations sur le profil de l'agent (voir Partie 4.2.3). Le discriminateur juge donc également si le message correspond au profil indiqué.

Enfin, pour rendre les approches évoquées précédemment encore plus efficaces, l'astuce du double-générateur est parfois utilisée (Cui et al., 2019 ; Olabiya et al., 2019b). En plus du générateur classique, qui génère une réponse en fonction d'une requête donnée, un deuxième générateur est ajouté pour générer les requêtes en fonction des réponses. Un discriminateur unique se charge donc de déterminer si les paires requête-réponse sont générées par des humains. Olabiya et al. (2019b) utilisent cette approche pour mettre en avant l'importance du profil, en ajoutant un générateur du profil en fonction de la réponse générée. Cui et al. (2019) démontrent que cette approche encourage les réponses plus diverses et spécifiques. En effet, les réponses génériques comme "Je ne sais pas" sont adéquates pour de nombreuses requêtes différentes. En revanche, les réponses plus diverses et spécifiques ne correspondent qu'à une requête précise. Cette architecture donne donc plus d'importance à la requête et décourage les réponses génériques.

## 4.5 Conclusion et tendances de la recherche en NLG

Sans être parfaitement exhaustif, cet état de l'art présente les approches neuronales majeures utilisées pour concevoir des systèmes de génération de texte. Toutes ces approches se différencient d'abord sur l'architecture sur laquelle elles se basent : soit un modèle de langue, soit un modèle Encodeur-Décodeur (voir Partie 3.2.2). Ces architectures permettent toutes les deux de générer du texte en fonction d'une séquence de mots en entrée. Leur différence réside principalement dans le fait que le modèle de langue produit les *tokens* un par un en fonction de tous les *tokens* précédents (ceux de l'input et de l'output en cours de génération), tandis que l'Encodeur-Décodeur encode d'abord la séquence d'entrée en une représentation vectorielle (l'*embedding*) qu'il décode ensuite pour produire une séquence de mots, à la manière d'un modèle de langue. Cette différence de stratégie, bien que subtile, peut avoir un impact sur la

compréhension du langage par le modèle et sa manière de générer du texte. Les deux approches ont leurs avantages. Néanmoins, on a vu que certains papiers (Mei et al., 2017 ; Zhang et al., 2018b ; Wolf et al., 2018) préconisent l'utilisation d'un modèle de langue dans le cadre d'une conversation : celui-ci serait plus efficace à comprendre l'évolution d'une conversation sur toute sa durée.

Au sein même de ces modèles, les approches se différencient par les techniques utilisées. Les architectures récurrentes de type RNN et LSTM, bien que très vieilles et largement éprouvées, sont encore utilisées dans de nombreux papiers à l'état de l'art. Les mécanismes d'attention se sont imposés comme indispensables pour comprendre ou générer du texte, d'abord comme ajout aux réseaux récurrents classiques, puis comme base de l'architecture Transformer. Cette architecture a progressivement remplacé ses prédécesseurs grâce à une meilleure capacité à utiliser l'information passée et une plus grande rapidité pour entraîner les modèles.

Avec le développement des techniques neuronales de NLG, des problématiques soulevées par les différents modèles ont poussé la recherche à se tourner vers une grande variété d'approches pour utiliser les modèles. On l'a vu, les systèmes conversationnels les plus classiques ont de nombreux défauts : réponses sans intérêt, répétitions, incohérences sémantiques et émotionnelles. Pour répondre à ces problèmes, de multiples solutions potentielles ont été étudiées, qui se résument souvent à conditionner le modèle sur de l'information extérieure à la conversation (profil de l'agent, informations factuelles, MN) ou même à changer complètement l'objectif d'optimisation, avec le *reinforcement learning* ou l'*adversarial learning*. Toutes ces approches répondent de manière intelligente et pragmatique au besoin d'améliorer des architectures de base, les modèles de langues et Encodeurs-Décodeurs, qui sont fonctionnelles mais loin d'être parfaites. Il est donc intéressant d'étudier ces approches pour comprendre les différents problèmes des systèmes de NLG et appréhender les approches prometteuses pour y remédier.

Malgré cette multitude d'approches différentes, une tendance se dessine clairement pour réaliser des systèmes de NLG plus performants : augmenter la taille des modèles et la masse des données sur lesquelles ils s'entraînent. Ces deux facteurs ont fait leurs preuves, notamment avec le succès de l'architecture Transformer et du *transfer learning* (voir Partie 4.1). Le Transformer, avec ses blocs encodeurs et décodeurs, peut facilement être agrandi en ajoutant simplement des blocs à la demande. Or, un théorème du DL est qu'un modèle plus grand nécessitera plus de données pour converger vers des paramètres optimaux. C'est là qu'entre en jeu le *transfer learning*, qui permet de pré-entraîner les modèles sur des quantités astronomiques de données, avant d'affiner les paramètres sur des données moins massives mais plus travaillées. Cette association fait qu'aujourd'hui les modèles les plus performants dans quasiment toutes les tâches de NLP sont les plus modèles comptant le plus grand nombre de paramètres. Cette propriété est d'ailleurs prouvée par l'expérience, avec notamment le modèle GPT-2 (Radford et al., 2019), publié successivement avec 124 Millions (M), 355 M, 774 M et 1,5 Milliards (Md) de paramètres, chaque version produisant de meilleurs résultats que les plus petites sur des tâches de NLP et NLG. Cette tendance est largement exploitée par les entités détenant les moyens techniques pour entraîner des modèles de cette taille. On a parlé de Meena (Google Research ; Adiwardana et al., 2020) et ses 2,6 Md de paramètres, Blender (Facebook AI Research ; Roller et al., 2020) et ses 9,4 Md, T5 (Google ; Raffel et al., 2019) et ses 11 Md, mais il y a aussi Megatron-LM (Nvidia ; Shoeybi et al., 2019) avec

8,3 Md, Turing-NLG de Microsoft et ses 17 Md de paramètres et le très récemment publié GPT-3 (Brown et al., 2020) avec un nombre ahurissant de 175 Md de paramètres. Ces modèles montrent des résultats impressionnants, s'approchant de plus en plus de textes produits par des êtres humains. Mais surtout, on observe que plus la taille du modèle augmente, plus ses résultats sont bons.

Néanmoins, entraîner et utiliser des modèles aussi gros nécessite d'énormes ressources de calcul que seuls les géants de l'industrie et de la recherche peuvent s'octroyer. Sans ces ressources, il faut donc toujours s'appuyer sur des approches plus pragmatiques qui permettent à des modèles de taille raisonnable de produire des résultats convaincants. Cela est d'autant plus vrai quand on cherche à construire des modèles destinés à être mis en production, comme ce sera éventuellement le cas pour nous. La mise en production amène son lot de contraintes, que ce soit au niveau technique qu'au niveau de la tâche en elle-même. En fonction de ces contraintes, différentes solutions pourront être étudiées. Dans la prochaine partie de cette étude, nous verrons quelles sont les contraintes de notre cas d'utilisation et quelles solutions ont été choisies pour construire notre système conversationnel.

## 5. Réalisation d'un chat-bot

Avec cet état de l'art des approches neuronales en NLG, nous avons une bonne vue d'ensemble des approches possibles pour construire un système conversationnel fonctionnel. Il nous faut maintenant bien définir notre tâche ainsi que les contraintes temporelles ou techniques qui l'accompagnent. Une fois cela réalisé, nous pourrions expliquer l'approche que nous avons choisie pour répondre au mieux à notre problème. Nous verrons ensuite les différentes étapes du développement et finirons avec de possibles suites au développement de notre chat-bot.

### 5.1 Définition de la tâche

Avant de commencer à développer notre chat-bot, commençons par définir clairement la tâche à réaliser. Tout d'abord, ce système conversationnel n'aura pas de tâche à réaliser dans la conversation : il sera dit “**open-domain**”, à l'inverse de “*goal oriented*” (voir Figure 1.1). Son seul objectif sera donc de converser, en anglais dans un premier temps, comme un être humain le ferait. Il n'a pas besoin d'être particulièrement informatif ou même intéressant. Il doit juste être capable de participer à une discussion avec un autre agent (humain ou artificiel), sans que l'on puisse détecter qu'il n'est pas humain. Pour cela, il doit d'abord générer du texte compréhensible et grammaticalement correct. Il doit également rester cohérent pendant toute la conversation. S'il se contredit ou s'il génère une réponse sans rapport avec la requête, il sera facilement identifiable comme artificiel. Enfin, encore dans l'optique de ressembler à un être humain, le chat-bot ne devra pas répéter des phrases, comme certains systèmes conversationnels le font (voir Partie 4.3).

Sur un plan technique, le modèle ne doit pas être trop gourmand en puissance de calcul. Déjà car nous devons être capables de l'entraîner, si besoin, et ce dans un temps raisonnable. Des modèles trop gros, c'est-à-dire qui comptent beaucoup de paramètres, vont demander beaucoup de mémoire vive pour être entraînés, et leur entraînement pourra durer jusqu'à plusieurs semaines ou mois, en fonction de la machine dédiée à ce processus. Il nous faudra donc choisir un modèle qui fonctionne sur notre matériel



informatique et qui pourra être entraîné pendant la durée de mon stage. Ensuite, sans entrer dans les détails de l'utilisation future de notre chat-bot, il faudra que le modèle soit utilisable sur des appareils équipés en composants classiques peu puissants. Là encore, un modèle trop gros nécessitera beaucoup de mémoire vive et des composants très efficaces pour pouvoir être utilisé avec un temps de réponse relativement court. Notre modèle doit pouvoir fonctionner sur des appareils avec 8 Go (Giga octets) de mémoire vive et avec un processeur classique (à l'inverse de graphique, voir Note 5.1).

#### Note 5.1 : CPU et GPU

Dans un ordinateur on trouve systématiquement un **processeur**, ou CPU (pour *Central Processing Unit* en anglais), qui se charge d'exécuter les instructions machines de tous les programmes informatiques. On peut parfois trouver aussi un **processeur graphique**, ou GPU (pour *Graphics Processing Unit* en anglais), qui s'occupe uniquement des calculs liés à l'affichage des programmes. Les GPUs sont très largement utilisés pour faire fonctionner des modèles de DL, car leur architecture est optimisée pour les calculs réalisés dans ces modèles. Ils permettent donc d'accélérer grandement l'utilisation de ces modèles.

## 5.2 Approche choisie

Pour construire notre chat-bot, il nous faut d'abord trouver un modèle de génération de texte, dans le cadre d'une conversation. Après notre étude de l'état de l'art il paraissait évident de choisir un modèle employant l'architecture Transformer, celle-ci donnant les meilleurs résultats de l'état de l'art dans pratiquement toutes les tâches du NLP. Des architectures étudiées dans la Partie 4.1, nous avons choisi d'utiliser **DialoGPT** (Zhang et al., 2019b) et ce pour plusieurs raisons. D'abord, le code du modèle, ainsi que ses paramètres entraînés, sont disponibles librement sur la plateforme de partage de code Github. Il est donc très simple de regarder comment est construit le modèle et de l'utiliser, sans même avoir à l'entraîner. De plus, DialoGPT utilise la librairie python `transformers` qui se veut très simple d'utilisation. Enfin, le modèle est disponible dans un format de 345 millions de paramètres. C'est un nombre certes relativement élevé qui nous donnera, on l'espère, de bons résultats. Mais il reste assez bas pour produire une réponse dans un temps limité sur une machine dotée de 8 Go de mémoire vive. DialoGPT est entraîné avec comme objectif la MLE. Notre premier besoin de générer du texte compréhensible et grammaticalement correct est donc rempli, car le modèle est entraîné à générer des phrases similaires à celles du corpus d'entraînement. Néanmoins, on a vu dans la Partie 4.3 que cet objectif apportait quelques problèmes, nous verrons plus bas comment nous les traitons.

Pendant la génération, plusieurs réponses possibles, appelées hypothèses, sont générées. À chacune d'entre elles est associé un score calculé par différents "**scoreurs**". Ces scoreurs sont des surcouches destinées à améliorer les réponses générées par DialoGPT. Tous les scoreurs associent à chaque hypothèse un score. On somme ensuite tous les scores de chaque hypothèse, et celle ayant le meilleur résultat est choisie comme réponse finale. Le premier scoreur est le **modèle backward** de DialoGPT. Comme expliqué dans la Partie 4.1, il tente de générer la requête en fonction de chaque hypothèse. Le score affecté à chaque hypothèse est le coût de chaque requête générée. Autrement dit, l'hypothèse qui a engendré la requête la plus proche de la vraie requête aura le meilleur score. Ce modèle *backward* nous permet donc de remplir notre objectif

de cohérence des réponses avec leur requête. Il nous permet également de produire des réponses moins génériques.

Un deuxième scoreur s'occupe d'assurer la cohérence avec les déclarations passées du chat-bot. Celui-ci est un modèle d'**implication textuelle**. Cette tâche de NLP consiste à lire deux phrases et à déterminer si elles sont en accord, si elles se contredisent ou si elles sont neutres. Nous avons donc choisi le modèle RE2 de Yang et al. (2019) pour réaliser cette tâche. Les phrases générées par le chat-bot sont gardées en mémoire dans un MN. Pendant la génération, pour chaque hypothèse générée, le modèle RE2 nous donne la probabilité que la phrase contredise chaque phrase présente dans le MN. On gardera la plus grande de ces probabilités. De ce fait, les hypothèses qui contredisent une ou plusieurs des phrases présentes dans le MN seront le plus pénalisées. Pour alléger le nombre de calculs à réaliser pendant la génération et améliorer les résultats de ce scoreur, seules les parties de phrases contenant potentiellement de l'information sont sauvegardées dans le MN. Pour cela, la librairie python `nltk` nous permet d'avoir le type de chaque mot : nom, adjectif, verbe à l'infinitif, verbe conjugué, participe passé, etc. Nous ne gardons seulement les parties de phrases qui ne sont pas des questions et qui sont de la forme "sujet-verbe-complément". Malheureusement, par manque de temps, ce scoreur sur l'implication textuelle n'a pas pu être implémenté complètement. Nous ne pourrions donc pas en vérifier l'efficacité.

Enfin, pour pallier le problème des répétitions de phrases, un troisième scoreur de **chevauchement** calcule la similarité de chaque hypothèse avec les dernières réponses générées. Un chevauchement est un groupe de mots présents dans l'hypothèse et une phrase précédemment générée. Pour chaque hypothèse, on calcule un score de chevauchement avec toutes les dernières déclarations du chat-bot. Ce score est calculé de sorte à pénaliser plus les grands chevauchements. Pour chaque hypothèse, on garde le pire score de chevauchement pour s'assurer qu'on ne génère pas une phrase trop répétitive.

### 5.3 Collecte de données

Après plusieurs essais d'interaction avec notre chat-bot, les échanges ont mis en évidence un manque de qualité des réponses générées par DialoGPT. Le modèle avait clairement subi un phénomène "*d'overfitting*" ("sur-apprentissage" en français), qui correspond au fait qu'un modèle apprenne "trop bien" à copier ses données d'entraînement, et peine donc à généraliser face à de nouvelles entrées. Concrètement, cela se manifestait avec des réponses peu variées et très proches des données d'entraînement de DialoGPT, à savoir un corpus extrait de discussions sur le forum Reddit.

La solution à ce problème était de **ré-entraîner** le modèle, ce qui nécessitait de trouver de nouveaux corpus de discussions. Nous avons donc commencé par télécharger tous les data-sets de dialogues en anglais disponibles. Ensuite, un premier tri a été fait pour retirer les corpus inutilisables dans notre cas : discussions avec trop de fautes d'orthographe ou trop ciblées sur une tâche précise et donc impossibles à comprendre sans le contexte de la tâche. Finalement nous avons gardé **17 data-sets**, résumés dans la Table 5.1. Un gros travail a ensuite été réalisé pour mettre tous ces data-sets sous le même format. Beaucoup d'entre eux avaient, en plus des dialogues, des informations supplémentaires spécifiques à leur tâche. Dans notre cas, nous n'avons besoin que des

déclarations des deux agents du dialogue. Un tri a également été réalisé pour retirer les discussions qui contiennent trop peu de déclarations (moins de deux déclarations), ou qui sont trop longues (plus de 1024 tokens), car notre modèle est limité dans la taille de ses inputs. Comme nous l'avons évoqué dans la Partie 4, certains data-sets comportent des informations sur le profil de chaque agent dans la conversation. Nous avons décidé de garder ces informations, quand elles sont disponibles, pour les utiliser si besoin dans le futur. Après nettoyage des données, il nous reste un total de **241 134 conversations**.

Nom	Référence	Tâche	Nombre de dialogues	Profil des agents
Blended Skill Talk	Smith et al., 2020	<i>Open-domain</i>	6,9k	Oui
Coached Conversational Preference Elicitation	Radlinski et al., 2019	Cinéma	502	-
Cornell Movie Dialogs Corpus	Danescu and Lee, 2011	<i>Open-domain</i>	45k	-
CraigslistBargain	He et al., 2018	Négociation	6,9k	-
DoQa	Campos et al., 2020	<i>Open-domain</i>	2,4k	-
Dream	Sun et al., 2019	<i>Open-domain</i>	6,4k	-
Empathetic Dialogue	Rashkin et al., 2018	<i>Open-domain</i>	25k	-
Frames	Asri et al., 2017	Réservations voyages	1,4k	-
MetalWOz	Shalymov et al., 2019	<i>Open-domain</i>	40k	-
MultiWOZ	Budzianowski et al., 2020	Réservations	10k	-
MuTual	Cui et al., 2020	<i>Open-domain</i>	8,8k	-
OpenDialKG	Moon et al., 2019	Culture	14k	-
Persona-Chat	Zhang et al., 2018a	<i>Open-domain</i>	11k	Oui
Soccer Dialogues	SmartDataAnalytics	Football	3k	-
Taskmaster 1 & 2	Byrne et al., 2019	Réservations	31k	-
Topical-Chat	Gopalakrishnan et al., 2019	<i>Open-domain</i>	11k	-
Wizard of Wikipedia	Dinan et al., 2018	<i>Open-domain</i>	22k	Un seul agent

Table 5.1 : Data-sets de dialogues choisis pour ré-entraîner notre système conversationnel.

## 5.4 Entraînement

Pour le ré-entraînement de DialoGPT, nous nous sommes basés sur l'approche de Wolf et al. (2019), détaillée dans l'article "*How to build a State-of-the-Art Conversational AI with Transfer Learning*" [7]. Dans cet article, Wolf décrit les étapes pour entraîner leur modèle de langue TransferTransfo. Comme on a vu dans la Partie 4.1, TransferTransfo utilise le même modèle de base pour tirer parti du *transfer learning* : GPT2 (Radford et al., 2018). Nous avons donc facilement pu adapter le script à notre modèle.

L'entraînement du modèle a été réalisé sur une machine dotée de puissants GPUs : des Nvidia Tesla V100 de 32 Go de mémoire vidéo. En utilisant quatre de ces GPUs, l'entraînement du modèle sur notre data-set pendant dix "époques" (on regarde toutes les données d'entraînement dix fois) a pu être fait en un peu moins de huit jours, à compter de dix-huit heures par époque.

## 5.5 Interaction avec le chat-bot

Une fois le modèle ré-entraîné, nous avons pu essayer notre prototype en discutant avec lui. Des captures d'écrans de ces interactions sont visibles dans l'Annexe 1. On voit que le modèle arrive très bien à tenir une conversation. Les phrases générées sont longues, bien formées et grammaticalement correctes. Le modèle arrive même à faire la conversation, en citant des éléments du monde réel, comme des films et des réalisateurs dans la première conversation. Néanmoins, en regardant de plus près, on peut voir que le modèle dit parfois des choses fausses. Dans la première conversation où nous discutons de cinéma, le modèle génère des phrases qui sont clairement fausses, comme le fait que le film "12 hommes en colère" ("*12 Angry Men*") n'est "pas si vieux que ça", ou que Richard Curtis est plus "*old school*" qu'Alfred Hitchcock. Plus loin il va même jusqu'à inventer le roman *Push* de Larry Kramer. Tous ces exemples montrent bien que le modèle est simplement entraîné à générer des phrases similaires à ce qu'il a vu dans son corpus d'entraînement, comme on l'a vu dans la Partie 4. On ne peut donc pas attendre de lui qu'il développe un vrai raisonnement sur des informations extérieures. On ne peut qu'espérer qu'il mette certaines informations dans le bon ordre, comme quand il évoque Winston Smith, le personnage principal du roman *1984*, lorsque j'évoque le livre en question.

Un autre défaut du modèle est apparu dans nos essais d'échanges : l'incohérence. Le modèle se contredit d'une phrase à l'autre et parfois même au sein d'une déclaration. Cela pourrait être réglé par le scoreur sur l'implication textuelle qui avait été prévu. Il aurait permis au modèle de vérifier que sa réponse ne contredit pas ses déclarations précédentes. Néanmoins, cette solution n'est pas parfaite, pour plusieurs raisons. D'abord, elle implique d'utiliser un autre modèle pour réaliser la tâche d'implication textuelle, et est donc gourmande en puissance de calcul. Ensuite, elle est dépendante de la qualité de ce modèle. Or, aucun modèle ne réalise cette tâche parfaitement. Enfin, la solution ne prend pas en compte la cohérence au sein même d'une déclaration. C'est donc une solution pratique, qui améliorerait les dialogues générés par le modèle, mais qui serait clairement imparfaite. Pour régler ce problème des incohérences, il faudrait s'intéresser davantage aux solutions étudiées dans la Partie 4.2 de l'état de l'art. En apportant des connaissances plus poussées au modèle sur le contexte de la conversation

ou sur des informations factuelles, on pourrait gagner en cohérence. Cela pourrait se faire par l'utilisation d'un MN plus sophistiqué que le nôtre, comme on a pu le voir dans la Partie 4.2.2. Néanmoins, l'incohérence des systèmes conversationnels reste un problème non-résolu et un sujet de recherche ouvert.

Cette phase de test de notre modèle nous a également permis de visualiser le gain en performances produit par notre ré-entraînement du modèle. L'Annexe 2 montre une conversation réalisée avec DialoGPT avant que nous ré-entraînions le modèle. Malgré une bonne base de conversation, avec des premiers échanges de qualité, DialoGPT était incapable de continuer un dialogue après huit échanges. Au-delà, il dégénérait et produisait des déclarations dépourvues de sens : mots ou lettres répétées, réponses vides ou phrases incohérentes. Après notre ré-entraînement du modèle, celui-ci produit des déclarations plus longues et mieux formées. Il peut également continuer des conversations sans jamais dégénérer. Cela met bien en avant l'importance d'entraîner ce type de modèle sur de très grandes quantités de données et de s'assurer de la qualité de ces données.

Enfin, nous avons également pu tenter de faire converser le modèle avec lui-même. En effet, comme nous l'avons dit plus haut, ce chat-bot aura pour but de dialoguer avec d'autres agents artificiels. Il fallait donc vérifier que le modèle reste bon quand il échange avec un autre bot. L'Annexe 3 montre un échange entre deux instances de notre prototype. Sans se concentrer sur le contenu de la conversation qui manque parfois de sens ou de cohérence, on voit que le modèle réussit à tenir la conversation.

## 5.6 Possibles développements futurs

Comme on vient de le voir, notre prototype a encore beaucoup de défauts. Il ne faut pas s'attendre à tous les régler, étant donné que la recherche sur les systèmes conversationnels est encore loin d'avoir abouti à un modèle parfait. Néanmoins, il reste beaucoup d'idées évoquées durant mon stage qui pourraient améliorer la qualité de notre prototype. Mon stage était malheureusement trop court pour pouvoir tout implémenter, mais j'ai gardé ces idées de côté en espérant qu'elles soient mises en œuvre dans le futur.

La première serait de simplement **ré-entraîner le modèle *backward*** de DialoGPT (voir Partie 5.2). Celui-ci est identique, dans son architecture, au modèle classique de DialoGPT qui nous sert à générer des réponses. Nous pourrions donc tirer parti des données que nous avons récoltées pour l'entraîner lui aussi. Cela aurait pour effet le rendre meilleur à générer la requête en fonction de la réponse, et donc de donner de meilleurs scores à chaque hypothèse générée.

Finir l'implémentation du scoreur sur l'**implication textuelle** serait aussi une solution simple et pratique pour améliorer la cohérence du modèle. Cela ne nécessiterait pas trop de travail, étant donné que le modèle est déjà entraîné et que le MN est implémenté. Il faudrait donc seulement intégrer le modèle au programme et ajouter le score d'implication textuelle à la *beam search*.

Le chat-bot Blender (voir Partie 4.1 ; Roller et al., 2020) implémente une idée simple qui pourrait améliorer les résultats de notre modèle. Les auteurs se sont aperçus que la longueur des réponses générées avait un fort impact sur la qualité des jugements humains du modèle. Les déclarations plus longues étaient considérées comme meilleures. Les auteurs ont donc contraint leur modèle à générer des déclarations plus

longues, en supprimant les *beams* trop courts, lors de la *beam search* (voir Note 4.2). C'est une astuce qui serait simple à implémenter et qui devrait, selon Roller et al. (2020), améliorer la qualité des réponses générées.

Enfin, une autre option serait tout simplement d'utiliser le chat-bot **Blender**. Comme nous l'avons dit dans la Partie 4.1, celui-ci a clairement montré les meilleurs résultats des modèles à l'état de l'art (sans considérer DialoGPT après notre ré-entraînement), et ce alors que nous n'avons essayé que le plus petit modèle qui compte 90 millions de paramètres, contre 345 millions pour la version de DialoGPT que nous utilisons. Deux raisons font que nous ne l'avons pas utilisé pour l'instant. D'abord, nous l'avons découvert bien plus tard que DialoGPT. Il était donc déjà question d'utiliser ce dernier. Mais surtout, Blender est bien plus complexe à utiliser dans notre cas. Pour l'essayer, nous avons dû passer par l'API (pour *Application Programming Interface*, voir lexique) ParlAI de Facebook qui regroupe tous les projets de NLP de l'entreprise. Cette API est pratique pour essayer différents modèles ou visualiser des data-sets. Mais elle devient très encombrante lorsque l'on veut se servir de ses modèles sans lui faire appel. En effet, dans notre cas nous avons besoin d'avoir un total contrôle sur le modèle, son architecture, ses paramètres et son utilisation. Nous ne pouvons donc pas passer par l'intermédiaire d'une API pour utiliser un modèle. Il y donc un travail à réaliser pour extraire Blender de ParlAI. Le code de ParlAI étant disponible librement sur internet, il est possible de retrouver le code de Blender et de l'extraire pour pouvoir l'intégrer à notre projet.

## E. Bilan et conclusion

### Ma mission

L'objectif de ma mission est rempli. L'état de l'art des approches neuronales permet d'avoir une idée des différentes approches possibles pour construire un système conversationnel. Sans être exhaustif, il cite les publications importantes et met en avant les approches moins orthodoxes. Il donne donc une bonne vue d'ensemble des différentes techniques permettant de construire un bon chat-bot.

Le prototype de chat-bot que je livre est fonctionnel. Comme on l'a vu, du travail pourrait encore être réalisé dessus pour améliorer ses résultats. Il ne sera de toute façon jamais parfait. Les systèmes conversationnels à l'état de l'art sont encore loin de pouvoir passer un vrai test Turing, à savoir tenir une conversation avec un être humain sans que celui-ci ne puisse déterminer que son interlocuteur est artificiel. Il faut donc trouver des astuces pour rendre des modèles imparfaits meilleurs. C'est ce que j'ai essayé de faire avec mon prototype. Celui-ci remplit sa tâche, puisqu'il est capable de prendre part à une conversation et générer des réponses similaires à celles qu'un humain pourrait produire.

J'espère donc que ce travail servira à mon équipe dans le futur pour leurs développements. Il est certain que le rapport sur l'état de l'art que je leur livre les aiguillera vers différentes approches, en fonction des tâches à réaliser. Le programme de mon prototype est écrit de manière à pouvoir ajouter des éléments facilement. Il leur permettra donc de continuer le développement, d'ajouter si besoin des couches supplémentaires pour améliorer le modèle, ou bien même de remplacer certaines parties pour suivre les dernières publications scientifiques à l'état de l'art.

### Mon expérience au ministère des Armées

Ce stage au ministère des Armées a été une expérience très enrichissante. Sur le plan technique, déjà. Dans ce rapport, j'ai essayé de retranscrire au mieux la très grande variété de sujets que j'ai appréhendé durant ce stage. J'avais déjà étudié la base des modèles de NLP pendant les cours de *Deep Learning* à l'ECE, mais rien ne vaut une expérience professionnelle pour réellement s'imprégner de ces notions techniques. D'abord parce que j'ai dû revoir toutes les notions que j'avais déjà étudiées pour m'assurer que je les comprenais parfaitement. Ensuite car je suis allé beaucoup plus loin dans mes recherches. Faire un état de l'art sur un sujet aussi large que les techniques de génération de texte m'a amené à lire des dizaines de publications scientifiques et à aborder des sujets auxquels je ne m'attendais pas, comme le RL ou l'AL. Et enfin parce que j'ai pu exercer mes compétences en programmation. J'ai appris à utiliser de nouvelles bibliothèques et découvert comment mettre en œuvre certains algorithmes de NLP que je n'avais jamais utilisés.

Le déroulement de mon stage a été dérangé par la pandémie de la **COVID-19**. Les mesures de confinement ont provoqué un chamboulement dans notre capacité à travailler. En effet, au ministère des Armées, il est normalement impossible de faire du télétravail. Heureusement, circonstances exceptionnelles obligent, cette politique a été allégée pour pouvoir permettre aux employés d'avancer sur leurs projets pendant les deux mois où le siège était fermé. En tant que stagiaire, notre travail n'étant pas confidentiel, il a donc été très simple de faire la transition pour travailler à la maison. J'ai

eu de la chance d'être en cours d'étude de l'état de l'art quand le confinement a débuté. Mon travail consistait donc à rechercher des publications scientifiques, les lire et en faire un résumé. J'étais donc parfaitement capable de continuer mon travail chez moi. Une fois l'état de l'art terminé, j'ai pu commencer à expérimenter sur les modèles de génération de texte en utilisant mon ordinateur personnel. Tout mon stage s'est déroulé en parfaite autonomie, le confinement n'a donc pas trop altéré ma manière de travailler. J'avais un rendez-vous téléphonique tous les mardis et jeudis matin avec mes deux tuteurs de stages. Ils pouvaient donc suivre mon avancement et je pouvais leur poser les questions que j'avais et m'assurer que la direction que je prenais leur convenait. Le confinement n'a donc pas été un problème pour moi. Ma mission n'a quasiment pas pris de retard et j'ai pu arriver au bout de mes objectifs.

Sur le plan strictement professionnel, ce stage m'a également beaucoup apporté. Pour ce dernier stage, mon but était d'avoir une nouvelle expérience en *Data Science*, en me rapprochant des métiers du *Deep Learning*. Le contrat est donc largement rempli. Avec ce stage, j'ai gagné à la fois des connaissances théoriques, avec l'état de l'art, et pratiques, avec la conception du chat-bot. Acquérir une telle expérience en NLP en seulement six mois est exceptionnel et cela me servira très certainement dans mes recherches d'emploi futures.

## **Mon futur**

Comme je l'ai dit dans l'introduction, l'objectif de ce stage de fin d'études était de trouver une mission de *Data Science* qui m'intéressait, dans une entreprise dans laquelle je pourrais avoir envie de travailler après mes études. Ce stage était donc exactement ce que je recherchais. Néanmoins, malgré le fait que le stage se soit parfaitement déroulé, j'ai décidé de ne pas faire la procédure de recrutement pour être embauché au ministère à la fin de mon stage. En effet, mon étude de l'état de l'art, pendant laquelle je me plongeais dans de nombreuses publications scientifiques, m'a fait découvrir une affinité particulière pour le monde de la recherche. Je me suis rendu compte que mon intérêt dans la *Data Science* était plus dans la théorie des algorithmes que dans leur implémentation. Cela faisait déjà longtemps que je savais cela, mais travailler sur l'état de l'art m'a permis de comprendre que cette idée correspondait au travail de chercheur. J'ai donc décidé de ne plus chercher de travail de *Data Scientist* et ai commencé à me tourner vers les laboratoires de recherche pour trouver un travail de chercheur ou un projet doctoral.

Ce stage m'aura donc surtout ouvert les yeux sur mes envies pour ma future carrière. Il m'aura permis de concrétiser des affinités que j'avais depuis longtemps, sans vraiment m'en rendre compte, avec le milieu de la recherche. C'est donc ce point, au-delà de toute autre considération technique ou professionnelle, que je retiendrai de ce stage.



# Lexique

**API** : *Application Programming Interface*, ou “interface de programmation” en français, ensemble de bibliothèques qui forment un outil auquel on peut faire appel dans un code informatique pour avoir accès à certaines fonctionnalités spéciales à cet outil.

**DL** : *Deep Learning*, ou « apprentissage profond » en français, discipline du ML impliquant des réseaux de neurones profonds : comportant plus de deux couches de neurones artificiels.

**Embedding** : représentation d’une variable discrète sous la forme d’un vecteur de nombres continus (voir Note 3.3).

**End-to-end** : “de bout en bout” en français, se dit d’un système qui réalise l’ensemble de la tâche d’une manière globale, à l’inverse des systèmes modulaires qui décomposent la tâche en plusieurs sous-problèmes traités séparément (voir Partie D.1).

**État caché** : dans un RNN, l’état caché (ou “*hidden state*” en anglais) est le résultat de la cellule récurrente à chaque étape du traitement. Il est utilisé pour calculer l’*output* du modèle à cette étape, et comme *input* du modèle à l’étape suivante du traitement.

**IA** : Intelligence Artificielle.

**Input** : données prises en entrée d’un modèle.

**LSTM** : réseaux *Long Short-Term Memory*, réseau récurrent (voir RNN) composé de portes permettant un meilleur transfert de l’information d’une étape à l’autre du traitement, et donc un meilleur développement de dépendances à long terme du modèle (voir Partie D.3.1.2)

**ML** : *Machine Learning*, ou « apprentissage automatique » en français, discipline de l’IA qui permet à des programmes informatiques d’apprendre à partir de données, en se basant sur des approches mathématiques et statistiques.

**MLE** : *Maximum Likelihood Estimation*, ou “estimation du maximum de vraisemblance” en français, technique d’optimisation de modèles qui cherche à les faire converger vers les outputs les plus probables (voir Partie D.4.3).

**MLM** : *Masked Language Model*, ou “modèle de langue masqué”, technique de pré-entraînement d’un modèle de langue qui consiste à masquer aléatoirement des mots dans les séquences d’un corpus. Le modèle s’entraîne à prédire les mots masqués (voir Partie D.4.1).

**MN** : *Memory Network*, ou “réseau à mémoire”, modèle utilisant une base de connaissances dynamique (voir Partie D.4.2.2).

**NLP** : *Natural Language Processing*, ou “traitement de langage naturel” en français, discipline de l’informatique qui rassemble les algorithmes de traitement automatique des données textuelles.

**NLG** : *Natural Language Generation*, ou “génération de langage naturel” en français, discipline de l’informatique qui cherche à générer du texte (voir Partie D.1).

**Open-domain** : “domaine ouvert” en français, se dit d’un système conversationnel qui n’a pas de but particulier dans la conversation, son seul objectif est de ressembler le plus possible à un être humain.

**Output** : résultats des calculs réalisés par un modèle.

**RL** : *Reinforcement Learning*, ou “apprentissage par renforcement” en français, théorie du **ML** consistant à faire apprendre une tâche à un agent en le faisant répéter différentes actions, pour lesquelles il gagne des récompenses (positives ou négatives). Le but de l’agent est de maximiser ces récompenses (voir Partie D.4.3).

**RNN** : *Recurrent Neural Network*, ou réseau de neurones récurrents en français, réseau de neurones comportant des liaisons récurrentes, c’est-à-dire dont l’*output* sert d’*input* à l’étape suivante du traitement (voir Partie D.3.1.1).

**RST** : *Rhetorical Structure Theory*, ou “théorie de la structure rhétorique” en français, théorie de la NLG qui définit une organisation du texte en décrivant des relations entre différentes parties de celui-ci (voir Partie D.2).

**Seq2Seq** : pour “*Sequence-to-Sequence*”, modèles prenant en entrée une séquence de données et générant une autre séquence en sortie (voir Partie D.3.2).

**T5** : *Text-To-Text Transfer Transformer*, modèle de NLP proposé par Raffel et al. (2019) (voir Partie D.4.1).

**Token** : “morceau” d’une phrase traitée par un modèle de NLP. Ça peut être une partie de mot, un mot, un signe de ponctuation ou un *token* spécial (voir Note 3.2).

# Ressources

## Sources :

- [1] *The Ultimate Guide to Recurrent Neural Networks in Python*, Nick McCullum : <https://www.freecodecamp.org/news/the-ultimate-guide-to-recurrent-neural-networks-in-python/>
- [2] *The Unreasonable Effectiveness of Recurrent Neural Networks*, Andrej Karpathy : <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- [3] *Illustrated Guide to LSTM's and GRU's: A step by step explanation*, Michael Phi : <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- [4] *The Illustrated Transformer*, Jay Alammar : <http://jalammar.github.io/illustrated-transformer/>
- [5] Site et benchmark de GLUE et SuperGLUE : [www.gluebenchmark.com](http://www.gluebenchmark.com)
- [6] *Deep Learning for Computer Vision: Generative models and adversarial training (UPC 2016)* : <https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016>
- [7] *How to build a State-of-the-Art Conversational AI with Transfer Learning*, Thomas Wolf : <https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313>

## Publications scientifiques

- Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu and Quoc V. Le, *Towards Human-like Open Domain Chatbot*. 2020.
- Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang and Lili Mou, *Affective Neural Response Generation*. 2017.
- Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra and Kaheer Suleman, *Frames: A Corpus for Adding Memory to Goal-Oriented Dialogue Systems*. 2017.
- Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville and Yoshua Bengio, *An actor-critic algorithm for sequence prediction*. 2016.
- Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*. International Conference on Learning Representations, 2015.
- John A. Bateman, *Enabling technology for multilingual natural language generation : the KPML development environment*. Cambridge University Press, 1997.
- Regina Barzilay and Mirella Lapata, *Aggregation via set partitioning for natural language generation*. Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, 2006.

Yoshua Bengio, Patrice Simard and Paolo Frasconi, *Learning Long-Term Dependencies with Gradient Descent is Difficult*. IEEE Transactions on Neural Networks, 1994.

Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka, *A bottom-up approach to sentence ordering for multi-document summarization*. Information Processing & Management, 2010.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, *Language Models are Few-Shot Learners*. 2020.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan and Milica Gasic, *MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling*. 2020.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik and Kyu-Young Kim, *Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset*. 2019.

Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Deriu, Mark Cieliebak and Eneko Agirre, *DoQA -- Accessing Domain-Specific FAQs via Conversational QA*. 2020.

Shaobo Cui, Rongzhong Lian, Di Jiang, Yuanfeng Song, Siqi Bao and Yong Jiang, *DAL: Dual Adversarial Learning for Dialogue Generation*. 2019.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang and Ming Zhou, *MuTual: A Dataset for Multi-Turn Dialogue Reasoning*. 2020.

Andrew M. Dai and Quoc V. Le, *Semi-Supervised Sequence Learning*. Advances in Neural Information Processing Systems, 2015.

Cristian Danescu-Niculescu-Mizil and Lillian Lee, *Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs*. 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of NAACL-HLT, 2019.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli and Jason Weston, *Wizard of Wikipedia: Knowledge-Powered Conversational agents*. 2018.

Ondřej Dušek and Filip Jurčíček, *A Context-aware Natural Language Generator for Dialogue Systems*. 2016.

Albert Gatt and Emiel Krahmer, *Survey of the State of the Art in Natural Language Generation : Core tasks, applications and evaluation*. Journal of Artificial Intelligence Research, 2018.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih and Michel Galley, *A Knowledge-Grounded Neural Conversational Model*. Proceedings of the 32th AAAI Conference on Artificial Intelligence, 2018.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency and Stefan Scherer, *Affect-LM : A Neural Language Model for Customizable Affective Text Generation*. 2017.

Eli Goldberg, Norbert Driedger and Richard I. Kittredge, *Using natural-language processing to produce weather forecasts*. IEEE Intelligent Systems, 1994.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel and Dilek Hakkani-Tür, *Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations*. 2019.

He He, Anusha Balakrishnan, Mihail Eriv and Percy Liang, *Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings*. 2017.

He He, Derek Chen, Anusha Balakrishnan and Percy Liang, *Decoupling Strategy and Generation in Negotiation Dialogues*. 2018.

Sepp Hochreiter and Jürgen Schmidhuber, *Long Short-Term Memory*. Neural Computation, 1997.

Jeremy Howard and Sebastian Ruder, *Universal Language Model Fine-tuning for Text Classification*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer and Mike Lewis, *Generalization through Memorization : Nearest Neighbor Language Models*. 2019.

Satwik Kottur, Xiaoyu Wang and Vitor R. Carvalho, *Exploring Personalized Neural Conversational Models*. Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.

Irene Langkilde and Kevin Knight, *Generation that exploits corpus-based statistical knowledge*. Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, 1998.

Mirella Lapata, *Automatic Evaluation of Information Ordering : Kendall's Tau*. Computational Linguistics, 2006.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, Bill Dolan, *A Diversity-Promoting Objective Function for Neural Conversation Models*. 2015.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao and Bill Dolan, *A Persona-Based Neural CONversation Model*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao and Dan Jurafsky, *Deep Reinforcement Learning for Dialogue Generation*. 2016b.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter and Dan Jurafsky, *Adversarial Learning for Neural Dialogue Generation*. 2017.

William C. Mann and Sandra A. Thompson, *Rhetorical Structure Theory : Toward a functional theory of text organization*. Text & Talk, 1988.

Hongyuan Mei, Mohit Bansal and Matthew R. Walter, *Coherent dialogue with attention-based language models*. Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.

Seungwhan Moon, Pararth Shah, Anuj Kumar and Rajen Subba, *OpenDialKG: Explainable Conversational Reasoning with Attention-based Walks over Knowledge Graphs*. 2019.

Oluwatobi Olabiyi, Anish Khazane, Alan Salimov and Erik T. Mueller, *An Adversarial Learning Framework For A Persona-Based Multi-Turn Dialogue Model*. 2019b.

Oluwatobi Olabiyi, Alan Salimov, Anish Khazane and Erik T. Mueller, *Multi-turn Dialogue Response Generation in an Adversarial Learning Framework*. 2019a.

Prasanna Parthasarathi and Joelle Pineau, *Extending Neural Generative Conversational Model using External Knowledge Sources*. 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever, *Improving Language Understanding by Generative Pre-Training*. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever, *Language Models are Unsupervised Multitask Learners*. 2019.

Filip Radlinski, Krisztian Balog, Bill Byrne and Karthik Krishnamoorthi, *Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences*. Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharam Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu, *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019.

Hannah Rashkin, Eric M. Smith, Margaret Li and Y-Lan Boureau, *Towards Empathetic Open-Domain Conversation Models: a New Benchmark and Dataset*. 2019.

Ehud Reiter and Robert Dale, *Building Applied Natural Language Generation Systems*. Cambridge University Press, 2000.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau and Jason Weston, *Recipes for building an open-domain chatbot*. 2020.

Iulian V. Serban, Alessandro Sordani, Yoshua Bengio, Aaron Courville and Joelle Pineau, *Building end-to-end dialogue systems using generative hierarchical neural network models*. Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016.

Igor Shalyminov, Sungjin Lee, Arash Eshghi and Oliver Lemon, *Few-Shot Dialogue Generation Without Annotated Data: A Transfer Learning Approach*. 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper and Bryan Catanzaro, *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2019.

Eric M. Smith, Mary Williamson, Kurt Shuster, Jason Weston and Y-Lan Boureau, *Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills*. 2020.

Kaito Song, Xu Tan, Tao Qin, Jianfeng Lu and Tie-Yan Liu, *MASS: Masked Sequence to Sequence Pre-training for Language Generation*. Proceedings of the 36th International Conference on Machine Learning, 2019.

Robyn Speer, Joshua Chin and Catherine Havasi, *ConceptNet 5.5: an open multilingual graph of general knowledge*. Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.

Vidhushini Srinivasan, Sashank Santhanam and Samira Shaik, *Natural Language Generation Using Reinforcement Learning with External Rewards*. 2019.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi and Claire Cardie, *DREAM: A Challenge Dataset and Models for Dialogue-Based Reading Comprehension*. 2019.

Zhiliang Tian, Wei Bi, Xiaopeng Li and Nevin L. Zhang, *Learning to Abstract for Memory-augmented Conversational Response Generation*. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng and Dongyan Zhao, *How to Make Context More Useful ? An Empirical Study on Context-Aware Neural Conversational Models*. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin, *Attention is All you Need*. Advances in Neural Information Processing Systems 30, 2017.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy and Samuel R. Bowman, *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 33rd Conference on Neural Information Processing Systems, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy and Samuel R. Bowman, *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2018.

Jason Weston, Sumit Chopra and Antoine Bordes, *Memory Networks*. 2014.

Michael White and Rajakrishnan Rajkumar, *Perceptron reranking for CCG realization*. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009.

Thomas Wolf, Victor Sanh, Julien Chaumond and Clément Delangue, *TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents*. 32nd Conference on Neural Information Processing Systems, 2018.

Jiawei Wu, Xin Wang and William Yang Wang, *Self-Supervised Dialogue Learning*. 2019.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou and Wei-Ying Ma, *Topic aware neural response generation*. Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.

Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, Haiqing Chen, *Simple and Effective Text Matching with Richer Alignment Features*. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

Tom Young, Erik Cambria, Iti Chaturvedi, Minlie Huang, Hao Zhou and Subham Biswas, *Augmenting End-to-End Dialog Systems with Commonsense Knowledge*. Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela and Jason Weston, *Personalizing Dialogue Agents : I have a dog, do you have pets too ?*. 2018a.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiuju Li, Chris Brockett and Bill Dolan, *Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization*. Advances in Neural Information Processing Systems 31, 2018b.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu and Bill Dolan, *Dialogpt: Large-Scale Generative Pre-training for Conversational Response Generation*. 2019b.

Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu, *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. 2019a.

Yinhe Zheng, Guanyi Chen, Minlie Huang, Song Liu and Xuan Zhu, *Personalized Dialogue Generation with Diversified Traits*. 2019.

Peixiang Zhong, Di Wang and Chunyan Miao, *An Affect-Rich Neural Conversational Model with Biased Attention And Weighted Cross-Entropy Loss*. Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu and Bing Liu, *Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory*. 2017.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu and Xiaoyan Zhu, *Commonsense Knowledge Aware Conversation Generation with Graph Attention*. Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018.



# Annexes

## 1. Conversations entre moi (human) et notre prototype de chat-bot (dialogpt) après ré-entraînement du modèle :

```
Turn 0 :
Agent #1 (human) >> Hi, what's your favorite movie?
Agent #2 (dialogpt) >> Hi, I really like the movie Inception. And my favorite movie is Star Wars, how about you?
Turn 1 :
Agent #1 (human) >> I love both of these films, but my favorite is 12 angry men.
Agent #2 (dialogpt) >> Oh, I like the movie 12 Angry Men, too. My friends were shocked, when I told them I love the movie.
Turn 2 :
Agent #1 (human) >> Why is that, it's so great! A bit old I guess but the story is so cool.
Agent #2 (dialogpt) >> Well, I guess just because it's not that old. Who wrote it?
Turn 3 :
Agent #1 (human) >> I think it's Hitchcock
Agent #2 (dialogpt) >> I love Hitchcock. He's my favorite director.
Turn 4 :
Agent #1 (human) >> He is very good, I like Birds and Vertigo.
Agent #2 (dialogpt) >> Birds is really good. Do you have a favorite movie of Hitchcock's?
Turn 5 :
Agent #1 (human) >> I like North by Northwest.
Agent #2 (dialogpt) >> That's a good one. I really liked The Lady Vanishes and Rear Window.
Turn 6 :
Agent #1 (human) >> Yeah they are all so good ! Who's your favorite director ?
Agent #2 (dialogpt) >> I like ones that are a bit more old school. My favorite director is Richard Curtis.
Turn 7 :
Agent #1 (human) >> Richard Curtis is not that old school...
Agent #2 (dialogpt) >> Yes, but he did use a lot of unconventional methods to his stories.
Turn 8 :
Agent #1 (human) >> I didn't know that, I guess I'll watch more of his movies. Which one do you recommend?
Agent #2 (dialogpt) >> I haven't seen too many of his movies, but if he is anything like the novel Push by Larry Kramer I'd recommend that one.
Turn 9 :
Agent #1 (human) >> I don't know this book. Is it good?
Agent #2 (dialogpt) >> It's about a young Russian girl who gets tangled in a web. It's really good, one of the best novels I've read recently.
Turn 10 :
Agent #1 (human) >> What other book do you recommend?
Agent #2 (dialogpt) >> If you enjoy reading novels you might try The Handmaid's Tale. It was adapted into a film.
Turn 11 :
Agent #1 (human) >> I heard about it, who wrote it?
Agent #2 (dialogpt) >> You have a great imagination. Lem Follet wrote it.
Turn 12 :
Agent #1 (human) >> No it's Margaret Atwood.
Agent #2 (dialogpt) >> No, your right, she wrote The Handmaid's Tale. You might try reading The Bonfire of the Vanities.
Turn 13 :
Agent #1 (human) >> Is it good ?
Agent #2 (dialogpt) >> Yes, it's very good and very entertaining. Do you like dystopian novels?
Turn 14 :
Agent #1 (human) >> Yes! I read 1984, I loved it, did you?
Agent #2 (dialogpt) >> Yes, I absolutely love 1984. The main character is Winston Smith. Are you familiar with his exploits?
Turn 15 :
Agent #1 (human) >> I am. But the story doesn't end well for him...
Agent #2 (dialogpt) >> That could be said about any of the "Big Mac" stories...
Turn 16 :
Agent #1 (human) >> I'm not familiar with thoses.
Agent #2 (dialogpt) >> Ok then. What other dystopian novels are you familiar with?
Turn 17 :
Agent #1 (human) >> I liked Fahrenheit 451!
Agent #2 (dialogpt) >> That was a great book. Did you know that Stephen King was terrified when he read the book Bambi, calling it the first horror movie he had ever seen.
```

```
Turn 0 :
Agent #1 (human) >> Hi! I've just passed my driving license!
Agent #2 (dialogpt) >> Awesome! Congratulations! Is it hard to get your license?
Turn 1 :
Agent #1 (human) >> Well yes a bit. You have to study the code and drive a lot.
Agent #2 (dialogpt) >> I don't have to study too much! Is it difficult to get a learners permit?
Turn 2 :
Agent #1 (human) >> No, that is simple.
Agent #2 (dialogpt) >> Cool. Do you know any interesting fact about cars?
Turn 3 :
Agent #1 (human) >> Yes, did you know there is a real car in space right now?
Agent #2 (dialogpt) >> I have not heard of this car but I think it's a space ship. Is this true?
Turn 4 :
Agent #1 (human) >> No I assure it's a real, drivable car.
Agent #2 (dialogpt) >> That's good. I like cars. My favorite is the Ferrari. I wish I had one.
Turn 5 :
Agent #1 (human) >> Yes Ferrari is great. What colour would you prefer?
Agent #2 (dialogpt) >> Red. What color is Ferrari? White or red?
Turn 6 :
Agent #1 (human) >> Usually it's red.
Agent #2 (dialogpt) >> Okay. Then I'll probably go with white. Where can I have a test drive of Ferrari?
Turn 7 :
Agent #1 (human) >> I don't know. I guess you can go on a track.
Agent #2 (dialogpt) >> Maybe if I go to Barcelona. I'd like to go to a track one day to get an idea of it. Where should I go and how should I start?
Turn 8 :
Agent #1 (human) >> I don't really know sorry. Do you like Aston Martins ?
Agent #2 (dialogpt) >> No I don't but that's cool they make racecars. And I do like them. Is that a track?
```

2. Conversation entre moi (User) et DialoGPT (Bot), avant que nous ré-entraînions le modèle :

```
User:Hello
Bot: What's up, What's up, What's up, What's up?
User:I'm working on a new book.
Bot: I can't wait!
User:Do you like to read ?
Bot: yes. but also I don't want to read.
User:What's your favourite book ?
Bot: I'm a really big Lovecraft fan.
User:Me too! I love The Call of Chtulhu.
Bot: Yeah it's a great story!
User:What are you up to ?
Bot: Nothing right now. I have a midterm tomorrow.
User:You should study!
Bot: No classes tomorrow!
User:Yes but you have a midterm
Bot: Okay...
User:Good
Bot: go get more
User:More what ?
Bot: Moo
User:Do you mean like a cow ?
Bot: Ooooooooooooo
User:I see
Bot: o
User:Interesting.
Bot: No whatever
```

### 3. Conversation entre notre prototype de chat-bot (dialogpt) et lui-même :

```
Init >> My back hurts so much.
Turn 0 :
Agent #1 (dialogpt) >> I'm sorry. I know how that's feeling. I suffered with chronic back pain for several years.
Agent #2 (dialogpt) >> I think chronic pain is something that a lot of people suffer with, but that doesnt mean it's a good thing. I hope it gets better for you.
Turn 1 :
Agent #1 (dialogpt) >> Thank you. Yes, it does get better. Have you heard of Chronic Pain Syndrome? It's an autoimmune disease.
Agent #2 (dialogpt) >> Oh i have! its awful. Is it caused by something outside of your body?
Turn 2 :
Agent #1 (dialogpt) >> No, it is caused by a combination of genetic and environmental factors.
Agent #2 (dialogpt) >> I see. I thought it had something to do with an autoimmune disease, i am glad you figured it out.
Turn 3 :
Agent #1 (dialogpt) >> It may be from an autoimmune disease, but most doctors don't know until people are in a really serious pain.
Agent #2 (dialogpt) >> I guess that is true for most conditions, but i just hope it isnt genetic, i would hate for something to happen to me.
Turn 4 :
Agent #1 (dialogpt) >> That would be awful. I'm pretty sure it's genetic, but I could be wrong.
Agent #2 (dialogpt) >> Im sure you're right! genetic disorders can be life threatening but there are a lot of medications to help with it.
Turn 5 :
Agent #1 (dialogpt) >> I know! I wish there were more effective drug companies in the world to help people with genetic disorders
Agent #2 (dialogpt) >> I agree. I would love to help someone with a genetic disorder, but not with a drug like anavar exposure.
Turn 6 :
Agent #1 (dialogpt) >> I agree, my friend Anavar causes a very unpleasant reaction. What is it to you
Agent #2 (dialogpt) >> It can cause red, itchy eyes, itchy stomach, shortness of breath, and nausea. My friend has it too. She needs lots of therapy.
Turn 7 :
Agent #1 (dialogpt) >> Oh, that must be so difficult to deal with. I hope with all the therapy she gets, she'll be fine
Agent #2 (dialogpt) >> It's worth it sometimes. She has a lot of friends in her profession so they can help as well. I wish each one of them could visit her.
Turn 8 :
Agent #1 (dialogpt) >> I can't even imagine! I hope each of them can go visit her.
Agent #2 (dialogpt) >> They can. I'm sure they are happy when her visits end.
Turn 9 :
Agent #1 (dialogpt) >> Mine don't end too often, but when they do, it is usually the happiest times of our lives.
Agent #2 (dialogpt) >> That is so true. Mine don't usually visit, but when they do, it is the best.
Turn 10 :
Agent #1 (dialogpt) >> I love when they do! Mine recently had an anniversary, and our whole family came over to celebrate.
Agent #2 (dialogpt) >> I bet they have a blast. That is such a sweet thing that they do.
Turn 11 :
Agent #1 (dialogpt) >> They do and I love spending time with them. Grandkids are fun too.
Agent #2 (dialogpt) >> My granny used to spend time with my kids too. She passed away a couple years ago.
Turn 12 :
Agent #1 (dialogpt) >> Sorry to hear that. Mine used to spend time with me as well. They are good kids.
Agent #2 (dialogpt) >> I think that they are. My grandkids are the best.
Turn 13 :
Agent #1 (dialogpt) >> Mine are good, too. I have grandchildren, but I still spend a lot of time with my cats.
Agent #2 (dialogpt) >> My cats also keep me busy. I love my cats, but I can't have them around the house.
Turn 14 :
Agent #1 (dialogpt) >> Cats are great. I'm glad you like them.
Agent #2 (dialogpt) >> I do like cats. I have one, but then I got another one.
Turn 15 :
Agent #1 (dialogpt) >> Oh. I've got a tabby cat. They can be a handful.
Agent #2 (dialogpt) >> Mine is a tabby and the most loving cat ever. Very sweet and lovely. What is your cat's name?
Turn 16 :
Agent #1 (dialogpt) >> Oh! I don't like the name for her. My cat's name is Cookie! I have 3 and they are all very sweet and cuddly.
Agent #2 (dialogpt) >> Cookie is a sweet name! I have always wanted one. Are they all females or Males?
Turn 17 :
Agent #1 (dialogpt) >> Mostly Males! They are always with me. I live alone so they have had plenty of time to get used to the name, but they are so excited to be with me!
Agent #2 (dialogpt) >> I see - I have two boys and two girls - they definitely get along great with my boys. It is a unique name, that is for sure!
Turn 18 :
Agent #1 (dialogpt) >> Yes that is unique. I have three son from one girl and three from another. Each one is a unique person lol
Agent #2 (dialogpt) >> Wow! I bet. What is your boy's name? Mine is Francis. Do yours all get along very well?
Turn 19 :
Agent #1 (dialogpt) >> Yes, they seem to. My daughter's name is Mina. My son's name is Dexter. My brother's name is George. LOL. Mine is the complete opposite - I have a boy and boy.
Agent #2 (dialogpt) >> My twin is always referred to as my little bro, but my nephews are always called George, Dexter, Harry, Jack and Charlie
```