UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MODÈLE INTÉGRÉ POUR LA GÉNÉRATION DE DIALOGUE ET LA TRADUCTION AUTOMATIQUE À BASE D'ATTENTION

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE

PAR

BILLAL BELAININE

FÉVRIER 2023

REMERCIEMENTS

Je tiens à exprimer ici ma reconnaissance à toutes les personnes qui ont contribué de près ou de loin à cette thèse.

Tout d'abord, je tiens à remercier ma directrice de recherche, Professeure Fatiha Sadat, pour son aide précieuse, sa patience et son encouragement, mais surtout pour avoir cru en moi et m'avoir donné l'assurance de croire en moi-même. Ses références pertinentes et ses nombreux commentaires avisés ont grandement contribué à réaliser cette thèse.

Je tiens à remercier également mon co-directeur de recherche, Professeur Mounir Boukadoum, mon guide, pour son soutien étendu et sa patience tout au long de mon doctorat ; sans lui l'achèvement de cette thèse n'aurait pas été possible.

Je remercie le professeur Hakim Lounis pour sa direction et ses conseils durant les deux premières années, sans oublier les autres professeurs du département d'informatique de l'UQAM pour la qualité de leur enseignement lors de ma thèse.

Je tiens également à remercier tous mes amis qui m'ont apporté leur soutien moral et intellectuel tout au long de ces années d'études. Je n'oublie pas mes remerciements à l'enseignant Abdelmadjid Lahmar pour sa correction linguistique de la rédaction. Finalement, je voudrais exprimer toute ma reconnaissance à ma famille à qui je dois tout et sans lequel ce travail n'aurait jamais vu le jour.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ABB Abréviation

TALN Traitement Automatique de Langues Naturelles

TAN Traduction Automatique Neuronale

TAS Traduction Automatique Statistique

TER *Translation Error Rate* (Taux d'erreur de traduction)

AIML *Artificial Intelligence Markup Language*

NLU *Natural Language Understanding*

URL *Uniform Resource Locator* (Localisateur uniforme de ressource)

WER *Word Error Rate* (Taux d'erreur de mot)

ALICE. *Artificial Linguistic Internet Computer Entity*

RNN *Recurrent Neural Network (en français : Réseau de neurones récurrents)*

RNR *Réseau de neurones récurrents*

TALN *Traitement Automatique de Langues Naturelles*

BLEU *BiLingual Evaluation Understudy*

LSTM *Long-Short Term Memory (en français : Mémoire à long-court terme)*

GPU *Graphies Processing Unit (en français : Unité de traitement graphique)*

CBOW *Continuous Bag-Of-Words (en français : Sac de mots continus)*

NLP *Natural Language Processing (en français : Traitement automatique de langues naturelles)*

ADAM *Adaptive Moment Estimation (en français : Estimation adaptative du moment)*

WER *Word Error Rate (en français : Taux d'erreur de mot)*

MSE *Mean Square Error (en français : Erreur quadratique moyenne)*

HRED *Hierarchical Recurrent Encoder Decoder*

VHCR *Variational Hierarchical Conversation RNN*

VAE *Variational AutoEncoder*

HRAN *Hierarchical Recurrent Attention Network*

MHV *Mots Hors Vocabulaire*

ReCoSa *Relevant Contexts with Self-Attention*

RI *Recherche d'informations*

NLG *Natural Language Generation*

GAN *Generative Adversarial Network*

GPT *Generative Pre-trained Transformer*

BERT *Bidirectional Encoder Representations from Transformers*

ALBERT *A Lite BERT*

RoBERTa *A Robustly Optimized BERT Approach*

BLEU *Bilingual Evaluation Understudy*

PPL *Perplexity*

MDA *Multi-Dimensionnel Attention*

WER *Word Error Rate* (Taux d'erreur de mot)

METEOR *Metric for Evaluation of Translation with Explicite ORdering* (Métrique d'évaluation de traduction avec l'ordonnancement explicite)

URL *Uniform Resource Locator* (Localisateur uniforme de ressource)

ROUGE *Recall-Oriented Understudy for Gisting Evaluation*

DSM *Direct Space Matching* (Correspondance directe de l'espace)

GAN *Generative Adversarial Networks* (Réseaux antagonistes génératifs)

MDA *Multi-dimensional Attention* (Attention multidimensionnelle)

RÉSUMÉ

Les agents conversationnels, aussi appelés *systèmes de questions-réponses* ou *chat-bots*, jouissent d'une grande popularité. Ces agents informatiques peuvent simuler une conversation avec un ou plusieurs humains par voie textuel ou même vocal. Ainsi, ils peuvent accomplir des tâches impressionnantes pour la société moderne en offrant à l'usager de l'aide, du soutien, des directions, des instructions, des suggestions, ou tout simplement une conversation.

Depuis quelques années, l'usage du modèle encodeur-décodeur s'est imposé en étant une approche de choix pour leur mise en oeuvre et, dans le cadre de la conversation à plusieurs répliques entre un usager et un agent (dialogue multi-tours). Ces architectures neuronales hiérarchiques ont suscité beaucoup d'intérêts. Cependant, elles reposent sur une mémoire de contexte qui est réalisée à l'aide de Réseaux de Neurones Récurrents (RNR), ce qui limite leur performance pour les longues séquences d'entrée.

Cette thèse s'intéresse au développement d'architectures neuronales de type encodeur-décodeur pour les longues conversations textuelles multi-tours. Les modèles développés peuvent non seulement servir à la génération de réponses dans les dialogues multi-tours ; mais peuvent aussi servir à la traduction automatique de séquences de phrases.

Cela est rendu possible grâce à une nouvelle vision architecturale de la modélisation des agents conversationnels neuronaux.

Dans cette thèse, nous proposons un modèle encodeur-décodeur qui utilise l'auto-attention multidimensionnelle pour remplacer le concept hiérarchique utilisé auparavant, en utilisant une généralisation du modèle *Transformer*. L'introduction de la représentation multidimensionnelle permet de retracer les sources des mots générés lors de la conversation sans recourir à un contexte créé à l'aide de réseaux de neurones récurrents (RNR). Le modèle présenté est le premier dans le domaine de la conversation multi-tours qui n'utilise ni encodeur hiérarchique ni RNR, ce qui lui permet d'éviter les problèmes d'oubli des RNR et le potentiel de propager les erreurs avec amplification d'un décodeur hiérarchique (à cause du traitement séquentiel sous-jacent). En outre, le modèle neuronal développé possède une complexité computationnelle plus faible, ce qui lui confère plusieurs avantages en termes de besoins en mémoire et de vitesse de calcul.

La validation du modèle a été complétée par le biais de trois corpus de tailles différentes

et les comparaisons subséquentes avec l'état de l'art ont permis de montrer que notre modèle surclasse plusieurs modèles de référence en *Perplexity* dans la génération de réponses et en score *BLUE* pour une tâche de traduction automatique à base de réseaux de neurones.

Mots clés : Traitement du Langage Naturel, agent conversationnel, traduction automatique, chatbot, génération de réponses, apprentissage machine, auto-attention, attention multidimensionnelle.

INTRODUCTION GÉNÉRALE

Contexte

Ces dernières années, le domaine de traitement automatique du langage naturel (TALN, ou NLP en anglais) a connu des progrès considérables. Ce domaine de l'intelligence artificielle (IA) tente de simuler l'intelligence humaine en matière de capacités linguistiques (Verspoor et Cohen, 2013). Son but est d'étudier et développer des programmes informatiques qui peuvent réaliser des tâches liées au langage humain et à la communication inter-humaine ; il aide les ordinateurs à interagir avec les humains en lisant et en créant un discours naturel. Cependant, le langage catégoriel humain est différent du langage binaire des ordinateurs et une représentation numérique de ce langage est requise pour un traitement par ordinateur. Par ailleurs, comprendre le langage humain est une tâche difficile, car il s'exprime de différentes manières, rendant la conversation diversifiée et complexe. Outre le grand nombre de langues existantes, chaque langue possède son vocabulaire, ses règles et sa grammaire spécifique. Grâce aux techniques de TALN, il est maintenant possible pour les ordinateurs de traiter la parole (Zhou et Wang, 2018), de comprendre du texte (Jiang *et al.*, 2020), de reconnaître et d'exprimer les émotions (Belainine *et al.*, 2020a; Zhou et Wang, 2018). En conséquence, ces techniques ont conduit à une évolution de la communication entre les machines et les humains et donc l'évolution des agents conversationnels (Antol *et al.*, 2015; Weston *et al.*, 2015; Xiong *et al.*, 2016; Sankar, 2020).

Dans ce contexte, les agents conversationnels, aussi appelé *chatbots*, sont devenus très populaires. Ces outils informatiques utilisent l'intelligence artificielle pour amorcer le dialogue en langage naturel avec un ou plusieurs usagers humains de manière convi-

viale. Dans un scénario typique, l'usager utilisateur est invité à formuler une requête et l'agent l'interprète et y répond avec son modèle conversationnel (Schuetzler, 2015).

Il existe deux approches principales relatives à la génération automatique de réponses dans les dialogues : à domaine fermé et à domaine ouvert. La première cible des tâches spécifiques à des domaines donnés, comme répondre aux requêtes des clients sur les sites Internet d'entreprises, tandis que la seconde ne vise aucun domaine particulier. Les systèmes a dialogue ouvert sont généralement plus difficiles à mettre en oeuvre en raison de leur portée plus large et de la diversité des réponses qui peuvent être générées. En particulier, ils doivent avoir une compréhension très approfondie du contexte pour être efficaces (Huang *et al.*, 2020; Gao *et al.*, 2018a). Les architectures neuronales génératives sont apparues comme une approche de choix pour modéliser de tels systèmes, en grande partie grâce à l'architecture encodeur-décodeur hiérarchique (HRED (Sordoni *et al.*, 2015b; Serban *et al.*, 2016b) et aux variantes qui ont été proposées (Serban *et al.*, 2017c; Park *et al.*, 2018; Wang et Jiang, 2019).

Les architectures neuronales génératives sont basées sur l'apprentissage automatique (Sankar, 2020). Ainsi, les systèmes développés apprennent des conversations par l'intermédiaire de corpus de dialogues qui contiennent des conversations alternées entre deux ou plusieurs interlocuteurs, où chaque interlocuteur participe à la conversation avec des énoncés comme illustré par l'exemple de la figure 1.

L'architecture du modèle neuronal pour la tâche désirée dépend grandement de la structure des corpus utilisé (Sankar, 2020). Pour les tâches de traduction automatique, les corpus comprennent deux segments de texte parallèles, l'un dans la langue source et l'autre dans la langue cible, ce qui a mené à l'adoption d'une architecture encodeur-décodeur où l'encodeur crée une représentation interne du texte source et le décodeur en déduit la traduction dans la langue cible (Zanettin, 2014). Dans le cas des dialogues, le format complexe des corpus associés a forcé les architectes à adopter une approche

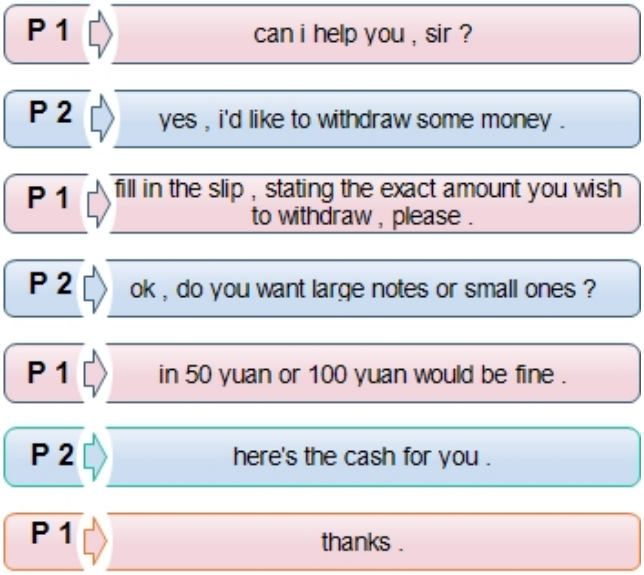


Figure 1: Exemple de conversations entre deux personnes tiré du corpus DialyDialog.

hiérarchique à encodage double (Sankar, 2020) : un premier encodeur encode les mots de chaque énoncé dans un vecteur de contexte qui est transféré au deuxième encodeur qui encode la suite des vecteurs obtenus afin de créer une représentation de l'historique de la conversation. Cette représentation sert alors de contexte global au décodeur qui l'utilise pour générer le prochain énoncé mot par mot (Li *et al.*, 2017b; Sankar, 2020; Zhang *et al.*, 2019a).

Jusqu'à récemment, le développement des architectures de traduction précédait celui de la génération de dialogues, à cause de la simplicité relative de la première tâche. Cela a mené à l'existence de deux architectures parallèles, la première applicable aux tâches de transduction et l'autre aux tâches de dialogue multi-tours. Cependant, une tâche de transduction est une sous-tâche de dialogue (dialogue à 1 tour). Donc, une architecture capable de générer des réponses pour les longs dialogues doit être capable de traduire une séquence en une autre séquence (Liu *et al.*, 2016; Hu *et al.*, 2020). Or, au regard des modèles proposés jusqu'à aujourd'hui, une recherche dans la littérature (voir Chapitre 2) montre qu'il existe peu, voir aucune recherche applicable au dialogue multi-tour et à

la traduction automatique en même temps ; ceci est l'une des motivations majeures du travail accompli dans cette thèse.

Ainsi, nous nous concentrons, dans cette thèse, sur les points suivants :

- amélioration du modèle neuronal pour la génération de dialogue ;
- amélioration du modèle neuronal en traduction automatique ;
- amélioration de la qualité et de la performance d'un modèle applicable pour les deux tâches.

Problématique de la recherche

Les agents conversationnels génératifs existants sont basés sur des architectures hiérarchiques qui utilisent une chaîne d'encodage composée de deux encodeurs en séquence et un décodeur. Le premier encodeur est responsable de l'encodage des mots de chaque énoncé en un vecteur de contexte, et le résultat est utilisé par le deuxième encodeur qui encode la séquence des énoncés, créant ainsi un contexte de dialogue qui est repris par le décodeur afin de générer sa réponse. (Zhang *et al.*, 2019a; Xing *et al.*, 2018).

La traduction en langage naturel est généralement considérée comme une tâche plus simple que la génération de réponses dans les systèmes conversationnels pour deux raisons principales (Liu *et al.*, 2016; Wei *et al.*, 2019; Sankar, 2020) :

1. Elle agit généralement sur une seule phrase d'entrée. À cet égard, il est concevable qu'un système capable de générer une réponse de dialogue soit adaptable à la traduction en langage naturel sans effort considérable.
2. Le nombre de choix pour traduire une phrase est limité en comparaison au nombre de réponses possibles à une question.

Lors de notre survol de la littérature, nous avons remarqué que tous les modèles d'encodage hiérarchique proposés sont loin de respecter la contrainte selon laquelle un sys-

tème capable de générer le dialogue est adaptable à la traduction, tel que les résultats de la traduction sont loin du résultat de base proposé dans l'état de l'art. Dans nos expériences sur la tâche de traduction automatique anglais-allemand pour les modèles hiérarchiques présentés dans le tableau 1, les meilleurs résultats obtenus ne dépassent pas un score Bleu de 12.11. Cependant, un modèle encodeur-décodeur non hiérarchique comme le modèle de base «séquence vers séquence» présenté par (Luong *et al.*, 2015a) a obtenu un score *BLEU* de 20.9 avec la même tâche de traduction. Cette grande différence des score révèle une limite de l'architecture hiérarchique.

	HRED	**VHRED**	**VHCR**	**HRAN**	**ReCoSa**
BLEU	11.43	10.68	10.36	11.94	12.11

Tableau 1: Score *BLEU* de différents modèles hiérarchiques pour la tâche de traduction automatique EN-GE (section 4.7)

De plus, plusieurs modèles à encodage hiérarchique (Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Park *et al.*, 2018; Xing *et al.*, 2018; Zhang *et al.*, 2019a) essaient de représenter la conversation humaine en utilisant deux types de séquences avec deux mécanismes d'encodage séquentiel comme suit : une séquence d'énoncés qui respecte un certain ordre d'alternance entre deux interlocuteurs et une séquence dans chaque énoncé contenant des mots bien ordonnés. Les ordres des séquences sont représentés soit de façon implicite avec des réseaux récurrents (Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Park *et al.*, 2018; Xing *et al.*, 2018), soit de façon explicite avec des transformations linéaires, dont celles utilisées par ReCoSa (Zhang *et al.*, 2019a). Ces deux types d'encodage rendent impossible la création de relations causales directes entre les mots sources et les mots de réponse à travers les attentions des deux encodeurs.

Dans la littérature sur les systèmes à encodeurs hiérarchiques, il n'existe pas non plus de mécanisme de traçage de contexte qui spécifie la source exacte de chaque mot généré dans la réponse à chaque énoncé de dialogue ainsi que dans chaque mots de l'énoncé

en même temps. En d'autres termes, une attention de dialogue long dépend des trois composants qui sont comme suit : la position du mot de sortie, la position du mot d'entrée et la position de l'énoncé. Ces trois composantes sont représentées dans une forme cubique dans un espace 3D. Un modèle comme ReCoSa (Zhang *et al.*, 2019a) nous fournit seulement l'attention entre les contextes de chaque énoncé avec les mots générés. D'autres modèles tels que les modèles séquence à séquence (Vinyals et Le, 2015; Sankar, 2020) avec attention créés pour la traduction automatique, nous fournissent une attention qui représente la relation entre les mots sources et les mots générés. Or, ce mécanisme de traçage peut aider à comparer les causalités source-cible entre les mots générés et ceux qui les ont précédés ainsi que les positions des énoncés dans le dialogue ou le texte source.

Contributions

Le travail décrit dans cette thèse propose une approche novatrice à la génération de réponses multi-tours et la généralisation d'une architecture de traduction automatique neuronale. Nous proposons un modèle qui peut faire les deux tâches du traitement automatique du langage naturel par le biais de trois contributions majeures, comme suit :

1. nous proposons une architecture encodeur-décodeur qui utilise un seul encodeur et un seul décodeur afin de minimiser les étapes de la chaîne d'encodage des énoncés et ainsi d'éliminer le potentiel de propagation et d'amplification d'erreurs entre les niveaux d'encodage ;
2. nous décrivons un mécanisme de traçage du contexte en spécifiant l'ordre des plongements des mots d'entrée par un double encodage de position, l'un pour leurs positions dans les énoncés respectifs et l'autre pour les positions des énoncés dans le dialogue ;
3. nous décrivons un mécanisme d'attention multidimensionnelle qui reflète les alignements de causalité entre les mots sources et les mots cibles sous forme de

cartes de chaleur. En d'autres termes, des causalités sémantiques seront générées au niveau du dialogue au complet, permettant ainsi de retracer des mots sources derrière chaque sortie générée à la fois, avec leurs positions dans les énoncés et leurs provenances, pour une meilleure prédiction et une meilleure interprétabilité du modèle ;

4. nous proposons une architecture neuronale qui permet autant la génération de réponses dans les systèmes de dialogues multi-tours, que la traduction automatique des séquences de phrases.

Dans ce contexte, notre recherche vise à créer un modèle neuronal pour la conversation multi-tours qui se caractérise par le remplacement de l'architecture hiérarchique par un mécanisme d'attention multidimensionnelle. De ce fait, le modèle possède aussi la capacité de s'adapter à des tâches simples du TALN comme la traduction automatique, en les considérant comme des dialogues à 1-tour.

Cette thèse comporte également la conception d'un modèle neuronal représentant un agent conversationnel multi-tours et son évolution à travers l'historique de la conversation. Ce modèle est capable de faire des tâches simples comme la traduction automatique ainsi que des tâches plus complexes tels que la générations des réponses.

Phases de la recherche

En vue d'atteindre nos objectifs, nous avons organisé notre recherche en plusieurs étapes comme suit :

La première phase est une tentative de création d'une nouvelle architecture encodeur-décodeur basée sur les modèles RNR. Dans cette architecture, nous nous sommes basés sur un seul encodeur et plusieurs décodeurs reliés entre eux par un mécanisme d'attention multidimensionnelle. Ce modèle est appelé le modèle en CASCADE.

La deuxième phase présente l'optimisation du modèle pour la traduction automatique.

Ce modèle est caractérisé par l'absence du module RNR. Il est basé sur une modification du modèle TRANSFORMER (Vaswani *et al.*, 2017) par laquelle l'attention clé-valeur de ce dernier est remplacée par une attention basée sur la similarité cosinus. Cette architecture nous a fourni de meilleurs résultats par rapport au modèle de base, tout en réduisant le nombre de paramètres.

La troisième phase présente une généralisation du modèle précèdent dans le but de créer un modèle de dialogue multi-tours. Notre contribution dans ce modèle est l'attention multi-dimensionnelle basée sur la similarité cosinus. L'architecture proposée nous a fourni de meilleurs résultats que ceux utilisant un concept hiérarchique de deux encodeurs, avec une nouvelle architecture basée sur un seul encodeur et un décodeur. De plus, ce modèle peut être appliqué à la traduction automatique comme déjà mentionné.

Hypothèses de la recherche

Quelques hypothèses ont guidé nos travaux. Tout d'abord, les recherches récentes on Intelligence Artificielle (IA) ont montré que les modèles hiérarchiques présentent une solution à la génération de réponses et peuvent améliorer la qualité des textes générés (Honghao *et al.*, 2017). Partant de cela, et en vue d'améliorer les modèles existants, nous nous sommes basés sur les hypothèses suivantes :

- Un bon modèle neuronal présenté pour la génération du dialogue multi-tours doit être capable d'accomplir plusieurs tâches du TALN, dont la traduction automatique.
- L'accès direct au contexte des mots générés par un seul encodeur est privilégié à l'accès indirect à travers un deuxième encodeur intermédiaire. Nous faisons l'hypothèse que nous pouvons avoir un modèle non hiérarchique basé sur le modèle encodeur-décodeur et qui améliore le résultat. Ainsi, il sera capable d'accomplir la tâche de la conversation multi-tours et la tâche de traduction automatique.
- Un modèle sans RNR comme ceux qui utilisent l'auto-attention (Kitaev *et al.*,

2020; Vaswani *et al.*, 2017) peut être généralisé pour une tâche de génération de dialogue. Ce modèle est parallélisable au niveau de l'application séquentielle des opérations avec une complexité de *O(1)* au lieu de *O(n)*.

Objectifs de recherche

L'objectif principal de notre recherche est de concevoir et de développer une architecture innovatrice d'agents conversationnels basée sur un modèle neuronal sans RNR. Nous visons dans un premier temps à définir un modèle qui simplifie le modèle Transformer (Vaswani *et al.*, 2017) pour la traduction automatique. Ensuite, ce modèle sera généralisé pour résoudre le problème de la conversation multi-tours. Le modèle général permettra de représenter les relations sémantiques entre les mots de la conversation et la réponse générée pour qu'il soit à la fois naturel pour l'humain et simulable par un agent de conversation.

Ce projet touche donc à deux applications du traitement automatique de la langue naturelle, soit la génération de réponses et la traduction automatique. Nous introduisons de nouveaux mécanismes permettant de :

- ⋆ Simplifier le modèle de l'auto-attention de manière à ce que le modèle de Transformer puisse se généraliser aux problèmes de la conversation multi-tours.
- ⋆ Modéliser l'abstraction de haut niveau de la conversation multi-tours en remplaçant le concept des modèles hiérarchiques par un modèle multi-attentionnel.

Afin de bien répondre aux problèmes présentés, il est important de décomposer l'objectif final en deux sous-objectifs indépendants pour valider les deux hypothèses l'une après l'autre :

1. Créer un traducteur neuronal sans utilisation de RNR basé sur le modèle Transformer (Vaswani *et al.*, 2017), la caractéristique de ce modèle est que son mécanisme attentionnel est simple et généralisable.

2. Créer le modèle généralisable sur le problème de conversation, en tenant toujours compte du fait que le modèle peut être appliqué à des modèles tels que la traduction automatique.

Sommaire et organisation de la thèse

Ce chapitre a présenté le contexte général dans lequel se situe cette thèse, ainsi que la problématique que nous tentons de résoudre, les hypothèses de recherche, et enfin, les objectifs que nous nous sommes fixés.

Le reste de cette thèse est organisé comme suit : le chapitre 1 couvre les concepts de base des réseaux de neurones ; par la suite, l'état de l'art y est présenté, en détaillant davantage les agents conversationnels, les corpus utilisés dans l'apprentissage des dialogues ainsi que toutes les approches de dialogue neuronal multi-tours. Le chapitre 3 propose la méthodologie de travail et le cadre expérimental sous forme d'une structure spécifique en trois parties. En outre, le chapitre 4 propose l'expérimentation de chaque partie du chapitre précédent et les résultats obtenus. Finalement, le chapitre 5 conclut notre travail en récapitulant les principales contributions et offre des perspectives de recherche future.

CHAPITRE I

CONCEPTS DE BASE SUR LES RÉSEAUX NEURONAUX

1.1 Introduction

Au cours des dernières décennies, les algorithmes d'apprentissage automatique classiques se sont révélés limités dans leurs capacités à traiter du texte brut en langage naturel (LeCun *et al.*, 2015). Les chercheurs ont consacré beaucoup d'efforts à la création de techniques où une représentation modifiée des données et des fonctionnalités appropriées permettent la prédiction de séquences ou le classement des données fournies en entrée. Cependant, cette ingénierie des fonctionnalités demande du temps et beaucoup d'expertise humaine pour concevoir des représentations de données efficaces.

Comme les performances des méthodes développées dépendent fortement de la phase de représentation des données ou de l'extraction des règles, et pour faciliter l'applicabilité des modèles d'apprentissage automatique, l'attention s'est tournée vers les techniques d'apprentissage profond (en anglais, deep learning) qui peuvent apprendre automatiquement les représentations et les règles sémantiques à partir des données (LeCun *et al.*, 2015). Ces représentations apprises automatiquement surpassent souvent les règles et les représentations de caractéristiques fabriquées à la main, et elles s'adaptent plus facilement aux nouvelles tâches sans exigence de l'intervention humaine.

Dans ce chapitre, nous présentons une revue de littérature et les concepts de base des

techniques de l'apprentissage profond dédiées au traitement du langage naturel, notamment les réseaux de neurones artificiels.

1.2 Les réseaux de neurones artificiels

Comme leur nom l'indique, ces outils de l'apprentissage automatique sont inspirés des réseaux de neurones biologiques et visent à imiter la façon dont les gens apprennent. Les réseaux de neurones artificiels apprennent les tâches en examinant des exemples sans être explicitement programmés, en tirant automatiquement un sens des représentations latentes des données non structurées. Ces représentations de haut niveau sont considérées comme trop complexes à capturer par les humains ou avec d'autres technologies informatiques (Sankar, 2020).

Les réseaux de neurones artificiels remontent à 1943 lorsque McCulloch et Pitts (1943) ont créé un modèle élémentaire du neurone humain, en tant que fonction linéaire à seuil. Cette fonction reçoit un ensemble des valeurs d'entrée $x_1, \cdots, x_n$ et les transforme en une sortie binaire y. Ainsi, le neurone de McCulloch et Pitts (1943) prédit deux sorties différentes en vérifiant si $y = f(x,w) = x_1 w_1 + \cdots + x_n w_n$ est positif ou négatif.

Dans les sous-sections suivantes, nous détaillerons des algorithmes de réseaux de neurones plus avancés.

1.2.1 Le neurone

Il s'agit de l'unité atomique de calcul des réseaux de neurones. Il prend comme entrée une représentation vectorielle incluant un ensemble de variables $X = \{x_1, \cdots, x_n\}$ et produit une seule sortie y. Dans les années 1950, l'algorithme *Perceptron* (Rosenblatt, 1958) a été le premier modèle capable d'apprendre les poids $\{w_1, \cdots, w_n\}$ à partir des exemples d'entrées de chaque catégorie. Le modèle peut être considéré comme un

classificateur binaire qui relie son entrée x à une seule valeur binaire $f(x)$.

$$f(x) = \begin{cases} 1 \text{ si } w \,.\, x + b < 0 \\ 0 \text{ sinon} \end{cases} \tag{1.1}$$

où $w.x = \sum_{i=1}^{n} w_i x_i$ où n désigne le nombre de variables d'entrée et b représente un biais. En 1960, les neurones linéaires adaptatifs (*ADALINE*) (Widrow et Lehr, 1990) ont apporté une amélioration au modèle précédent en prédisant un nombre réel par l'ajout d'une fonction non linéaire. Plus précisément, le neurone prend un vecteur d'entrée X à n dimensions et l'associe à un vecteur de poids W et un vecteur de biais B. La sortie neuronale est représentée par (LeCun *et al.*, 2015) :

$$f(x) = \sigma(\sum_{i=1}^{n} w_i x_i + b) \tag{1.2}$$

Où σ est une fonction d'activation non linéaire qui transforme l'espace vectoriel d'entrée.

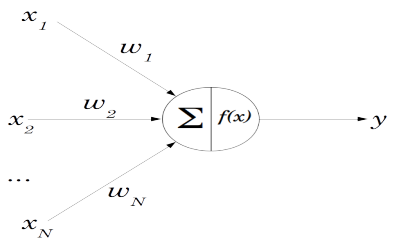


Figure 1.1: L'architecture d'une unité neuronale.

1.2.2 Réseau de neurones à action directe (Feed Forward)

Un réseau neuronal à action directe est un réseau multicouche où les sorties des neurones de chaque couche sont transmises aux neurones de la couche suivante. Ce réseau est composé de trois types de couches : couche d'entrée, couche cachée et couche de sortie. Chaque couche est entièrement connectée avec la couche précédente et la couche

suivante, ce qui signifie que chaque neurone d'une couche prend en entrée toutes les sorties de la couche précédente. De plus, il n'y a pas de liaison de connexion entre les unités d'une même couche. Les unités de la couche d'entrée sont des valeurs scalaires tandis que les unités de la couche cachée correspondent aux unités neuronales, calculant une somme pondérée de leurs entrées puis appliquant une fonction d'activation non linéaire. Formellement, la sortie de la couche cachée est la suivante :

$$h = f(Wx + b) \tag{1.3}$$

Où f est une fonction d'activation non linéaire comme $tanh$ ou $sigmoide$, $x \in R^{d_{in}}$ est un vecteur de nombres réels représentant l'entrée in, avec d_{in} le nombre de dimensions d'entrée. $b \in R^{d_h}$ dénote un biais et $W \in R^{d_h \times d_{in}}$ représente la matrice de poids apprise pendant l'entraînement (LeCun *et al.*, 2015).

La couche de sortie calcule une sortie finale basée sur le vecteur de représentation $h \in R^{d_h}$. Cette sortie peut être un nombre réel ou une distribution probabiliste à travers des mots de vocabulaire, dépendant de la tâche du réseau. De même que la couche cachée, la couche de sortie a une matrice de poids U avec souvent pas de vecteur de biais. Le réseau multiplie la matrice de poids U par le vecteur caché h afin de générer une sortie finale z comme suit :

$$z = Uh \tag{1.4}$$

Où $z \in R^{d_{out}}$, d_{out} étant le nombre d'unités de sortie out, et $U \in R^{d_{out} \times d_h}$. Si la tâche à accomplir représente la classification, la sortie est un vecteur à encodage chaud ou de probabilités. Dans le premier cas le plus grand élément est mis à 1 et les autres à 0 (one-hot encoding en anglais) et dans le second, les éléments sont normalisés à des valeurs comprises entre 0 et 1 avec un total de 1. Une fonction qui réalise cette normalisation est la fonction $Softmax$ (Goodfellow *et al.*, 2016) definie par :

$$Softmax(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{d_{out}} exp(z_j)}, 1 \leq i \leq d_{out} \tag{1.5}$$

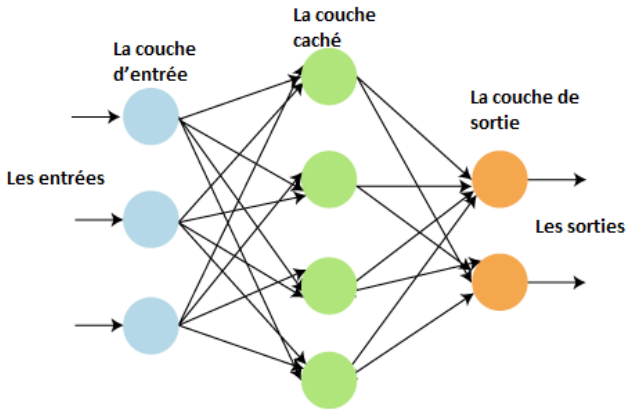


Figure 1.2: Architecture d'un réseau de neurones à action directe avec une couche cachée.

1.2.3 Entraînement d'un réseau de neurones

Cette opération vise à trouver les poids optimaux qui minimisent la distance moyenne, appelée fonction d'erreur, entre la sortie $\hat{y}$ produite par le modèle pour une entrée x et la sortie y souhaitée pour la même entrée. L'appellation fonction de perte est aussi utilisée pour refléter le fait que la fonction d'erreur peut être définie de différentes façons. La fonction d'erreur la plus populaire est l'erreur quadratique moyenne (Mean Square Error - MSE) entre $\hat{y} = f(x)$ et y. Cette fonction est utilisée pour résoudre les problèmes de la régression et elle est définie comme suit (Sankar, 2020) :

$$J(\theta) = -\mathbb{E}[\left\|y_j - \hat{y}\right\|^2] \tag{1.6}$$

Où θ fait références à tous les paramètres de réseau comme la matrice des poids $W^k \in \mathbb{R}^{d_k X d_{k-1}}$ ou un biais $b^k \in \mathbb{R}^{d_k}$ d'un couche k du réseau. $\mathbb{E}$ représente la moyenne.

Pour les classificateurs probabilistes, la fonction de perte couramment utilisée est la probabilité de négatif du log-vraisemblance (*Negative Log Likelihood*) J qui garantit qu'une probabilité maximale est attribuée aux bons résultats et une probabilité minimale est attribuée aux mauvais résultats. La perte J est définie comme suit (LeCun

et al., 2015) :

$$J(\theta) = -\sum_{j=1}^{|V|} y_j log\hat{y} \tag{1.7}$$

Où $\hat{y}$ représente le vecteur de probabilité prédite, y est le vecteur de vérité terrain ainsi que $|V|$ est la taille du vocabulaire ou le nombre des classes prédites.

Notre objectif est de trouver les paramètres θ qui minimisent cette fonction :

$$\hat{\theta} = \underset{\theta}{arg\ min}\ J(\theta)$$

Les valeurs des paramètres θ du minimum de la fonction $J(\theta)$ correspondent aux valeurs θ qui peuvent rendre le dérivée de la fonction $J(\theta)$ égale à zéro. Les algorithmes d'optimisation tels que la descente de gradient stochastique (Robbins et Monro, 1951) ou Adam (Hoang et Kang, 2019) pourraient être utilisés. Ces algorithmes cherchent à trouver le minimum d'une fonction en identifiant dans quelle direction la pente de la fonction augmente le plus fortement puis en se déplaçant dans la direction opposée (Noseworthy, 2018; Goodfellow *et al.*, 2016) (voir figure 1.3).

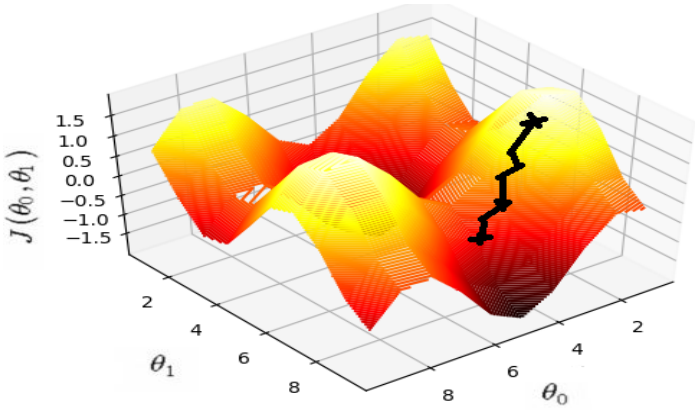


Figure 1.3: Illustration de minimum de la fonction de perte, après le déplacement de θ dans le sens opposé à la pente de la fonction avec un taux de déplacement est η (Tang *et al.*, 2015; Goodfellow *et al.*, 2016).

Dans le descente de gradient, le mouvement vers le minimum recherché est paramétré par un coefficient η appelé taux d'apprentissage. La valeur de η doit être réglée avec soin, car l'apprentissage prendra trop de temps à converger si elle est trop petite, et les mises à jour des poids peuvent dépasser le minimum et diverger si elle est trop grande. Les paramètres θ du modèle sont ainsi mis à jour comme suit (Noseworthy, 2018) :

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta^{(t)}} J(\theta^{(t)}) \tag{1.8}$$

où $\nabla J(\theta^{(t)})$ est le vecteur de gradient.

L'algorithme de rétro-propagation (Hinton *et al.*, 1986) utilise une règle de différenciation en chaîne pour calculer le gradient de toutes les couches du réseau, incluant les couches cachées dont on ne connaît pas les sorties désirées à priori. Cela est fait en prenant la dérivée partielle de la fonction de perte $\nabla_{\theta^{(t)}} J(\theta^{(t)})$ par rapport à chaque paramètre du modèle. L'algorithme fonctionne comme suit :

1. Propager l'entrée via le réseau.
2. Calculer la moyenne de la perte globale pour un bloc de données.
3. Calculer les gradients et mettre à jour les pondérations de la couche de sortie.
4. Propager l'erreur en arrière et mettre à jour les poids des couches d'entrées et cachées.
5. Répéter avec les données d'apprentissage par lots. Si la fonction de perte J est dans la tolérance d'erreur, terminer. Sinon, passer à une autre époque (c'est-à-dire un passage complet à travers l'ensemble de données) (Goodfellow *et al.*, 2016).

1.3 Apprentissage profond pour le traitement du langage naturel

Le traitement automatique du langage naturel (TALN) est défini comme l'analyse et l'utilisation du langage humain par la machine. Il aide les ordinateurs à interagir avec les humains en lisant et en créant un discours naturel. Cependant, le langage catégoriel

humain est différent du langage binaire des ordinateurs et une représentation numérique de ce langage est requise pour un traitement par ordinateur. Par ailleurs, comprendre le langage humain est une tâche difficile, car il s'exprime de différentes manières, rendant la conversation diversifiée et complexe. Outre le grand nombre de langues existantes, chaque langue a son vocabulaire, ses règles et sa grammaire spécifique. Grâce aux techniques de TALN, il est maintenant possible pour les ordinateurs de traiter la parole, de comprendre du texte, de reconnaître et d'exprimer les émotions (Belainine *et al.*, 2020c,a,b).

De nombreuses applications de TALN sont basées sur des modèles de langage qui calculent une distribution de probabilités sur des séquences des mots et des caractères. L'apprentissage profond est un ensemble des techniques des réseaux de neurones artificiels qui ont été appliquées avec succès dans divers domaines tels que les tâches de TALN, allant du traitement de la parole jusqu'à l'analyse sémantique (Hinton *et al.*, 2012; Iyyer *et al.*, 2014). Parmi les tâches les plus difficiles dans le domaine du TALN figurent :

- La traduction automatique (Bahdanau *et al.*, 2015; Koehn *et al.*, 2007; Papineni *et al.*, 2002; Li *et al.*, 2016a,b; Sordoni *et al.*, 2015b).
- L'analyse sémantique qui étudie la signification des mots, simples ou composés, ainsi que les rapports de sens entre les mots. (Jiang *et al.*, 2020; Deerwester *et al.*, 1990; Belainine *et al.*, 2020c).
- Les résumés automatiques (Erkan et Radev, 2004; Nallapati *et al.*, 2016; Rush *et al.*, 2015).
- Les questions/réponses (Antol *et al.*, 2015; Weston *et al.*, 2015; Xiong *et al.*, 2016).

Afin d'obtenir d'excellentes performances dans les tâches de TALN, les mots à traiter doivent d'abord être représentés par des vecteurs denses avant l'utilisation comme entrées du modèle neuronal. Le processus correspond à l'apprentissage de vecteurs de

plongement pour chaque mot cible.

Pour représenter des mots sur des vecteurs de nombres réels, les vecteurs de plongement des mots nous permettent d'effectuer une certaine notion de similitude (par exemple, distance cosinus, distance euclidienne, etc.) (Mikolov *et al.*, 2013; Pennington *et al.*, 2014).

1.3.1 Le plongement de mots (word embedding)

Le plongement de mots, appelé aussi plongement lexicaux, est considéré de nos jours comme la méthode de choix pour capturer de nombreuses relations sémantiques latentes entre les mots à partir d'un corpus non étiqueté. L'idée d'apprendre une représentation distribuée et continue a été présentée pour la première fois par Bengio *et al.* (2003) qui ont montré qu'un représentation vectorielle offre la possibilité d'un mécanisme d'encodage numérique des mots tout en reflétant leurs proximités sémantiques. L'intuition est que les mots ayant des significations similaires ont tendance à apparaître les uns à côté des autres dans le texte et qu'ils auraient donc des représentations vectorielles similaires dans l'espace. Le modèle neuronal apprendra un plongement en initialisant aléatoirement le réseau, puis le déplacera de manière itérative pour ressembler aux vecteurs de mots proches. *word2vec* (Mikolov *et al.*, 2013) et *GloVe* (Pennington *et al.*, 2014) sont parmi les méthodes les plus populaires pour une telle représentation. *word2vec* présente deux approches qui apprennent les vecteurs des mots à travers un réseau neuronal à anticipation afin de prédire les mots de voisinage. *Skip-Gram* fonctionne pour prédire les mots de contexte basés sur le mot central *Continuous Bag Of Words (CBOW)* fait le contraire. Bien que la méthode *word2vec* réussit à capturer des relations sémantiques complexes, elle ne parvient pas à utiliser les statistiques de cooccurrence et de fréquence complètes (Pennington *et al.*, 2014). *GloVe* contourne ce problème en définissant un modèle des moindres carrés pondérés (*WLS-Weighted least squares*) qui s'entraîne sur le nombre de cooccurrences mot à mot, utilisant ainsi les

statistiques des fréquences du corpus (Pennington *et al.*, 2014).

1.3.2 Modèle de langue

La modélisation du langage (*Language modeling (LM)*) consiste à prédire les prochains mots en utilisant les mots précédents comme contexte. Ce modèle estime une distribution du texte en langage naturel en attribuant des probabilités aux séquences de mots, avec des valeurs plus élevées pour les phrases grammaticalement correctes et très fréquentes dans un corpus. Par exemple, dans un corpus où la phrase « *apprentissage profond* » est très fréquente, une probabilité plus élevée est assignée au mot « *profond* » avec chaque apparition de mot « *apprentissage* ».

Formellement, étant donné une phrase $s = (w_1, \cdots, w_m)$ de longueur m, le modèle de langue définit une probabilité $P(s)$ en prédisant individuellement chaque mot dans la séquence compte tenu de tous les mots précédents. En utilisant la règle de probabilité en chaîne, le calcul de la probabilité conjointe d'une séquence $P(w_1, w_2, \cdots, w_m)$, peut se faire comme suit :

$$P(s) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_m|w_1\cdots w_{m-1}) \tag{1.9}$$

où $P(w_m|w_1, \cdots, w_{m-1})$ représente la probabilité de prédire un mot w_m parmi tous les mots précédents $(w_1, \cdots, w_{m-1})$.

Cependant, il n'est pas facile de calculer la probabilité conditionnelle d'un mot en tenant compte de tous les mots qui le précèdent lorsque la chaîne est longue. Afin de contourner ce problème, un modèle de n-gramme a été proposé dans le but de raccourcir l'historique des mots à seulement les N derniers mots. Par exemple, le modèle bigramme simplifie la probabilité $P(w_m|w_1\cdots w_{m-1})$ et utilisant uniquement la probabilité conjointe avec le mot précédent, ce qui conduit à l'hypothèse de Markov (Mikolov *et al.*, 2011) :

$$P(w_m|w_1\cdots w_{m-1}) \approx P(w_m|w_{m-1}) \tag{1.10}$$

Nous estimons la probabilité conditionnelle $P(w_m|w_{m-1})$ en calculant le nombre du bi-gramme $w_m w_{m-1}$ et l'échelle du nombre unigramme pour le mot w_{m-1} :

$$P(w_m|w_{m-1}) = \frac{C(w_m w_{m-1})}{C(w_{m-1})} \tag{1.11}$$

Cette technique nommée *sac de mots*, place les mots dans un vecteur de vocabulaire qui peut devenir très grand en fonction de la taille du vocabulaire (Jurafsky et Martin, 2017).

La modélisation du langage N-gram est simple et rapide à mettre en œuvre, mais son inconvénient majeur est de ne pas pouvoir gérer la dépendance à long terme, ainsi que la généralisation au contexte long et aux mots hors vocabulaire (MHV) (Jurafsky et Martin, 2017), c'est la raison pour laquelle des réseaux récurrents ont été créés.

1.3.3 Réseaux de neurones récurrents

Les tâches du TAL comme la traduction automatique sont caractérisées par le fait que des données d'entrée possèdent un format séquentiel dont l'ordre des mots est très important. Les réseaux à action directe ne possèdent pas un mécanisme qui peut inclure l'ordre des mots. De plus, ces réseaux prennent en entrée des exemples de taille fixe et les documents ou phrases que l'on désire tester sont en général de tailles variables. Les Réseaux de Neurones Récurrents (RNR) ont été conçus pour fonctionner sur des séquences de données. Ils possèdent un mécanisme de gestion de mémoire dans le but de capter l'information contextuelle des états calculés antérieurement.

En théorie, les RNR peuvent mémoriser le contexte de longues séquences. Cependant, en pratique, ils ne comprennent que quelques étapes en arrière, car les valeurs du gradient évoluent de manière combinatoire lors de l'apprentissage (Bengio *et al.*, 2003). Ce problème est appelé le problème de fuite du gradient ou l'étranglement du gradient (Goodfellow *et al.*, 2016). Plusieurs architectures proposées ont essayé de contourner ce problème. Parmi ces architectures, la mémoire court et long terme (en anglais :

Long-Short Term Memory ou LSTM (Hochreiter et Schmidhuber, 1997)) et l'unité récurrente fermée (en anglais : gated recurrent unit GRU (Chung *et al.*, 2014)) qui fournissent de nouvelles façons de calculer les états cachés. Ces architectures sont capables de mieux capturer le contexte de longues séquences grâce à des portes introduites dans ces modèles. Le rôle principal des portes est de déterminer quelle information doit être gardée ou ignorée d'une itération à l'autre. Ces modèles réduisent le problème d'oubli des RNR sans l'éliminer complètement (Cheng *et al.*, 2016; Chung *et al.*, 2014; Jozefowicz *et al.*, 2015; Kalchbrenner *et al.*, 2016).

Un autre obstacle avec les RNR est lié au fait de traiter les séquences de phrases mot par mot. Cela peut affecter la complexité du modèle qui devient *O(N)* ; *N* étant la taille de la séquence (Vaswani *et al.*, 2017). De plus, bien que l'information passée est censée être conservée à travers les états cachés passés, les modèles de séquence à séquence suivent la propriété de Markov où chaque état est censé dépendre uniquement de l'état vu précédemment (Zhou *et al.*, 2020; Goodfellow *et al.*, 2016). Les RNRs introduisent aussi le concept de mémoire dans chaque unité, ce qui alourdit le modèle en termes de paramètres. À cet égard, un accès direct à l'information comme la solution introduite par le concept de l'attention (Bahdanau *et al.*, 2015; Luong *et al.*, 2015b) est toujours préférable à la réception d'une copie transformée de cette information (Wolf *et al.*, 2020).

1.3.4 L'attention dans l'approche générative

Afin de construire des systèmes de traduction automatique, on utilise souvent une architecture de type encodeur-décodeur. Le rôle de l'encodeur est d'apprendre la composition séquentielle d'une phrase source et celui du décodeur est de déduire celle de la phrase cible correspondante. Pour modéliser cette architecture dite séquence vers séquence (seq2seq), on utilise des modèles de langage comme les RNRs. Cependant, les chercheurs ont voulu aussi trouver un mécanisme qui identifie les mots d'une phrase

source qui ont le plus de relation avec chacun des mots à générer dans la phrase cible. C'est le mécanisme d'attention, introduit par, (Bahdanau *et al.*, 2015), puis affiné par (Luong *et al.*, 2015a). L'idée clé de ce mécanisme consiste à établir des connexions directes à court terme entre la cible et la source en accordant une attention particulière au contenu source pertinent pendant la traduction. Un sous-produit de ce mécanisme est la matrice de visualisation qui représente l'alignement entre la source et la cible (comme le montre la figure 1.4).

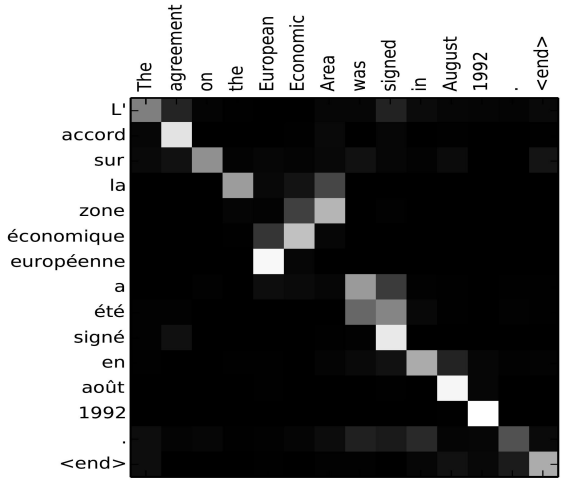


Figure 1.4: Visualisation de l'attention - exemple des alignements entre les phrases source et cible. La figure est tirée de (Bahdanau *et al.*, 2015).

Dans le modèle simple de séquence vers séquence qui utilise seulement deux réseaux récurrents, nous passons le dernier état de la source de l'encodeur au décodeur lors du démarrage du processus de décodage (Sankar, 2020). Cela fonctionne bien pour les phrases courtes et moyennes. Par contre, pour les phrases longues, l'état caché unique et de taille fixe se fait étrangler par le nombre de paramètres et la quantité d'information considérée durant le déplacement entre les états (la mémoire à court terme ne peut charger toute l'information de l'encodeur) (Luong *et al.*, 2015a; Sankar, 2020). Au lieu de rejeter tous les états cachés calculés dans le réseau récurrent source, le mécanisme

d'attention fournit une approche qui permet au décodeur d'y jeter un coup d'œil (en les traitant comme une mémoire dynamique de l'information source), et ainsi améliorer la génération des textes longs (Bahdanau *et al.*, 2015; Luong *et al.*, 2015a; Goodfellow *et al.*, 2016).

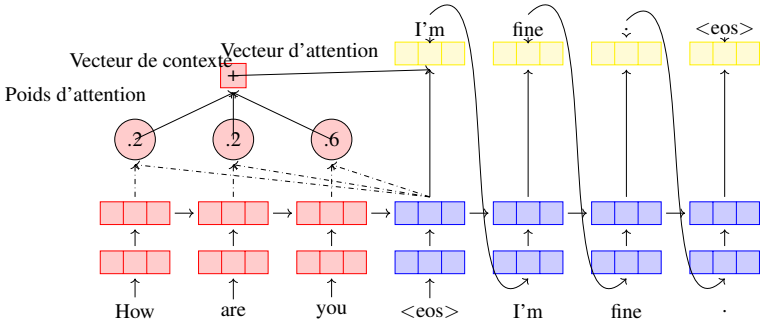


Figure 1.5: Architecture d'un système basé sur l'attention dans l'approche générative.

Comme illustré par la figure 1.6, le calcul d'attention se produit à chaque pas de temps du décodeur. Il comprend les étapes suivantes (Bahdanau *et al.*, 2015; Luong *et al.*, 2015a) :

1. L'état caché de la cible actuelle h_t est comparé à chaque état source $\overline{h_s}$ pour dériver un poids d'attention α_{ts} (peut être visualisé comme dans la figure 4.1).
2. Calcul d'un vecteur de contexte global c_t en tant que moyenne pondérée des états sources $\overline{h_s}$ sur la base des poids d'attention α_{ts}.
3. Combinaison du vecteur de contexte c_t avec l'état caché de la cible actuelle h_t pour obtenir le dernier vecteur d'attention a_t.
4. Le vecteur d'attention est alimenté comme une entrée à l'étape suivante du décodeur.

Les trois premières étapes peuvent être résumées par les équations ci-dessous :

Les poids d'attention : $\alpha_{ts} = \frac{exp(score(h_t, \overline{h_s}))}{\sum_{s'}^{S} exp(score(h_t, \overline{h_{s'}}))}$

Le vecteur de contexte : $c_t = \sum_s \alpha_{ts} \overline{h_s}$

Le vecteur d'attention : $a_t = f(c_t, h_t) = tanh(W_c[c_t; h_t])$

$$score(h_t, \overline{h_s}) = \begin{cases} v_a^T tanh(W_1 h_t + W_2 \overline{h_s}) \text{(Bahdanau } et\ al.\text{, 2015)} \\ h_t^T W \overline{h_s} \text{ (Luong } et\ al.\text{, 2015a)} \end{cases} \quad (1.12)$$

1.3.5 Transformer

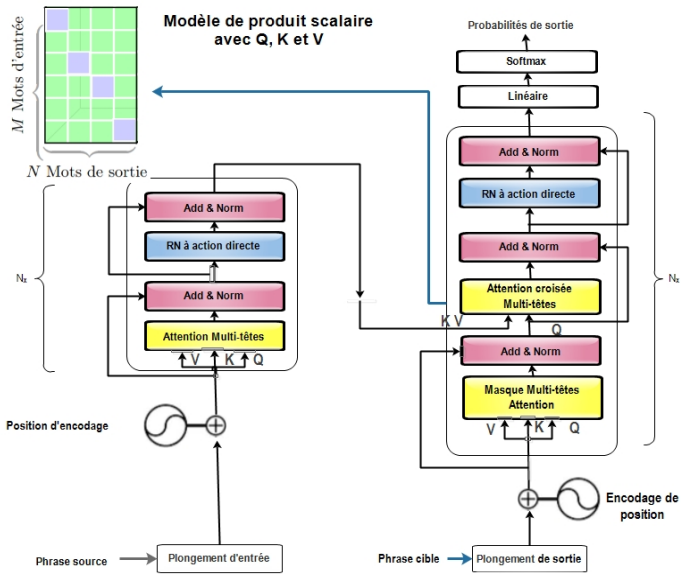


Figure 1.6: Illustration de l'architecture du Transformer (Vaswani *et al.*, 2017)

Le Transformer est un modèle d'apprentissage profond introduit en 2017 comme nouvelle architecture de transduction séquence à séquence, avec application au TAL (Vaswani *et al.*, 2017). Comme les réseaux de neurones récurrents (RNR), les Transformers sont conçus pour gérer des données séquentielles comme celles du langage naturel et des tâches comme la traduction automatique. Contrairement aux RNRs, les Transformers n'exigent pas de traitement ordonné des données. Par exemple, dans le cas où les données d'entrée représentent une phrase, le Transformer n'a pas à traiter le premier

mot de la phrase avant le dernier. Cette propriété permet une parallélisation du traitement beaucoup plus importante qu'avec les RNRs et donc des temps de formation réduits (Vaswani *et al.*, 2017).

Depuis son introduction, le Transformer est devenu le modèle de choix pour résoudre de nombreux problèmes du TAL, remplaçant les anciens modèles de RNR tels que le LSTM et le GRU. Comme le modèle facilite davantage la parallélisation pendant la formation, cela a permis l'entraînement avec des volumes de données jugés impossibles auparavant. Cette possibilité d'entraînement avec des données massives a conduit au développement des systèmes pré-entraînés tels que BERT (Bidirectional Encoder Representations from Transformers) et GPT (Generative Pre-Training Transformer), formés avec d'énormes ensembles de données en langage général et pouvant être affinés pour des tâches linguistiques spécifiques (Devlin et Chang, 2018; Radford *et al.*, 2019a; Devlin *et al.*, 2019; Brown *et al.*, 2020).

L'architecture du Transformer repose sur un mécanisme d'attention à base de produit scalaire implémenté sous forme « d'auto-attention». Le rôle de l'auto-attention est de remplacer la mémoire introduite dans le RNR. Lorsqu'une phrase est traitée, les poids d'attention sont calculés simultanément entre les vecteurs des mots d'une même phrase, produisant ainsi un contexte pour chaque mot introduit. Par conséquent, le vecteur du contexte produit ne contient pas seulement des informations sur le mot lui-même, mais aussi une combinaison d'informations d'autres mots pertinents qui sont pondérés par leurs poids d'attention.

Chaque unité d'auto-attention apprend trois matrices de poids ; les requêtes W_Q, les clés W_K et les valeurs W_V. Pour chaque mot d'indice i, mot_i, son plongement x_i est multiplié par chacune des trois matrices de poids afin de produire un vecteur de requête $q_i = x_i W_Q$, un vecteur clé $k_i = x_i W_K$, et un vecteur de valeur $v_i = x_i W_V$. Les poids d'attention sont calculés à l'aide de la requête et des vecteurs clés. Ainsi, pour chaque pair de mots

mot_i et mot_j, le poids d'attention a_{ij} est le produit scalaire entre q_i et k_j. Les poids sont divisés par la racine carrée de la dimension des vecteurs clés, $\sqrt{d_k}$ qui stabilise les gradients pendant l'entraînement et passe ensuite dans une fonction $Softmax$ qui les normalise pour former un total égal à 1.

La sortie de l'unité d'attention pour un mot_i est la somme pondérée des vecteurs de valeur de tous les mots pondérés par leurs poids a_{ij}.

Le calcul d'attention pour tous les mots peut être exprimé comme un grand calcul matriciel, ce qui est utile pour la formation, en raison de l'optimisation de calcul des opérations matricielles. Les matrices Q, K et V sont définies comme les matrices où les i-emme lignes sont des vecteurs q_i, k_i, et v_i respectivement, et on a :

$$\text{Attention}(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V(Vaswaniet\ al.,\ 2017) \tag{1.13}$$

Le Transformer est un modèle encodeur-décodeur dont chaque sous-module (encodeur ou décodeur) est constitué d'un empilement de couches unitaires. Une couche unitaire d'encodeur et de décodeur est construite sur la base des attentions. Notant ici que la couche unitaire du décodeur est composée d'une attention supplémentaire par rapport à la couche unitaire de l'encodeur nommée «attention croisée». Son rôle est de déterminer la relation entre les mots d'une phrase source représenté par les matrices K et V avec la phrase cible représenté par la matrice Q dans l'équation 1.13.

La fonction de chaque couche d'encodeur est de traiter son entrée pour générer des vecteurs d'encodages, contenant des informations sur les parties des entrées qui sont pertinentes les unes pour les autres. La couche de l'encodeur transmet ses vecteurs d'encodages de sortie à la prochaine couche en tant qu'entrée. Ensuite, chaque couche du décodeur prend tous les encodages et les traites en utilisant son auto-attention ainsi que l'attention croisée (Vaswani *et al.*, 2017; Wolf *et al.*, 2020).

Les couches d'encodeur et de décodeur ont d'autres modules supplémentaires comme

le réseau neuronal à action directe qui peut fonctionner comme une mémoire supplémentaire (Irie *et al.*, 2020), le module de normalisation par couche (Ba *et al.*, 2016) et le module des connexions résiduelles qui peut aider à garder le gradient le plus longtemps possible durant sa propagation dans les couches supérieures afin d'éviter sa disparition (He *et al.*, 2016).

1.3.6 L'encodage positionnel

Les réseaux de neurones récurrents (RNRs) prennent en compte implicitement l'ordre des mots d'une phrase traitée, puisqu'ils analysent ces mots en séquence. Ce n'est pas le cas du Transformer où il faut incorporer un sens d'ordre afin de spécifier la position de chaque mot dans la phrase avant les étapes d'encodage et de décodage. La méthode utilisée est nommée « Encodage positionnel » (Vaswani *et al.*, 2017; Wolf *et al.*, 2020).

Afin de trouver cet encodage, les critères suivants ont été posés (Vaswani *et al.*, 2017) :

1. Un encodage doit être unique et généré pour chaque pas de temps (la position du mot dans une phrase).
2. La distance entre deux pas de temps doit être cohérente entre les phrases de différentes longueurs.
3. Le modèle devrait se généraliser à des phrases plus longues et ses valeurs doivent être bornées.

L'encodage proposé par Vaswani *et al.* (2017) satisfait tous ces critères. Tous d'abord, il s'agit d'un vecteur de dimension d équivalant à la taille du vecteur de mots et donnant de l'information sur une position spécifique dans une phrase. Ensuite, ce vecteur est utilisé pour équiper chaque mot de l'information sur sa position dans la phrase avec des valeurs continues. En d'autres termes, nous améliorons l'entrée du modèle par l'injection de l'ordre des mots.

Soit pos la position d'un mot dans une phrase d'entrée $\overrightarrow{p_{pos}} \in \mathbb{R}^d$ est son vecteur d'en-

codage de position correspondant. La taille des dimensions d'encodage d est celle des vecteurs de mots, dont i représente une position dans le vecteur des mots. Ensuite, $F : \mathbb{N} \to \mathbb{R}^d$ est une fonction qui produit le vecteur de sortie et $\overrightarrow{p_{pos}}$ sera définie comme suit (Shiv et Quirk, 2019) :

$$\overrightarrow{p_{pos}}^{(i)} = f(pos)^{(i)} = \begin{cases} sin(\omega_i.pos) \text{ si i=2k} \\ cos(\omega_i.pos) \text{ si i=2k+1} \end{cases} \tag{1.14}$$

Où

$$\omega_i = \frac{1}{10000^{2k/d}} \tag{1.15}$$

Ce schéma d'encodage positionnel possède deux propriétés clés. D'abord, chaque position a un encodage unique d'un mot dans une phrase, permettant au modèle de s'occuper de n'importe quelle position absolue donnée. Ensuite, chaque encodage positionnel de x a une relation avec son encodage relative $x+t$, c'est-à-dire, chaque deux positions x et y peuvent être modélisées par une transformée affine $f(y = x+t)$ avec les encodages positionnels $f(x)$ et $f(t)$. On peut vérifier cette propriété par l'existence d'une transformation linéaire $M \in \mathbb{R}^{2x2}$, M étant indépendant de x, telle que (Shiv et Quirk, 2019) :

$$\begin{bmatrix} f(x+t)^{(2i)} \\ f(x+t)^{(2i+1)} \end{bmatrix} = \begin{bmatrix} sin(\omega.(x+t)) \\ cos(\omega_i.(x+t)) \end{bmatrix} = \begin{bmatrix} cos(\omega_i.(t)) & sin(\omega_i.(t)) \\ -sin(\omega_i.(t)) & cos(\omega_i.(t)) \end{bmatrix} \begin{bmatrix} sin(\omega_i.(x)) \\ cos(\omega_i.(x)) \end{bmatrix}$$

$$\begin{bmatrix} f(x+t)^{(2i)} \\ f(x+t)^{(2i+1)} \end{bmatrix} = \begin{bmatrix} f(t)^{(2i+1)} & f(t)^{(2i)} \\ -f(t)^{(2i)} & f(t)^{(2i+1)} \end{bmatrix} \begin{bmatrix} f(x)^{(2i)} \\ f(x)^{(2i+1)} \end{bmatrix} = M \begin{bmatrix} f(x)^{(2i)} \\ f(x)^{(2i+1)} \end{bmatrix}$$

1.3.7 La recherche en faisceau

Dans les modèles génératifs, de phrases comme séquence vers séquence, nous avons la distribution de chaque mot w_i de la phrase t conditionné par l'historique des mots

$w_1, \cdots, w_{i-1}$ et le contexte c. L'objectif est de trouver la phrase la plus probable $\hat{t}$ pour un contexte donné c, plus précisément :

$$\hat{t} = arg\ max\ \underset{t}{log}\ p_\theta(t|c) = \underset{w_1 \cdots w_n}{arg\ max} \sum_{i=1}^{n} log\ p_\theta(w_i|w_1 \cdots w_{i-1}, c)$$

La résolution de la séquence des mots la plus probable est coûteuse en mémoire, car, elle nécessite l'exploration de toutes les phrases générées, en utilisant tous les mots de vocabulaires dans chaque pas de la séquence. Pour cette raison, une inférence approximative est appliquée en utilisant l'algorithme de recherche en faisceau afin de décoder la phrase la plus probable (Graves, 2012).

La recherche en faisceau (*Beam-Search* en anglais) est une optimisation heuristique de l'algorithme de parcours en largeur appliqué aux graphes. Elle permet de réduire sa consommation en mémoire. Dans cet algorithme, seul un nombre prédéterminé de k meilleurs solutions sont explorés comme étant des candidats (Meister *et al.*, 2020).

L'algorithme conserve un faisceau de k phrases partielles les plus probables classées par leur log-vraisemblance. À chaque itération, chaque phrase partielle du faisceau est étendue en ajoutant chaque membre du vocabulaire V à la fin de la phrase partielle. Le log-vraisemblance de chacun des $|V|xk$ phrases est calculé et les k premiers mots sont conservés. Ceci est répété jusqu'à la fin de la séquence et renvoie la phrase la plus probable à la fin (Graves, 2012; Noseworthy, 2018).

1.4 Conclusion

Dans ce chapitre, nous avons donné un aperçu général sur les concepts de base des réseaux de neurones, ainsi que les composants unitaires et la méthode de formation de ces modèles.

Nous avons également présenté les concepts de base de différentes approches liées au modèle de langage et à la représentation du texte dans ce modèle. Nous avons aussi

expliqué comment les techniques de conception ont évolué au fil des ans : Le chemin de l'amélioration vers les réseaux de neurones récurrents et ses variantes telles que *LSTM* et *GRU* jusqu'à l'arrivée du Transformer.

Nous concluons que l'apprentissage profond a parcouru un long chemin en commençant par le modèle de langue standard, en passant par le modèle de langage neuronal récurrent et en finissant par les modèles basés sur l'auto-attention comme Transformer, BERT (Devlin *et al.*, 2019) et GPT (Brown *et al.*, 2020).

Dans le chapitre suivant, nous présentons une revue de la littérature qui examine l'état-de-l'art et les différentes stratégies utilisées dans la création d'un agent conversationnel, appelé aussi chatbot, ainsi que leurs différentes classifications. Une attention particulière sera portée à la présentation des agents génératifs multi-tours.

CHAPITRE II

ÉTAT DE L'ART

Dans ce chapitre, nous présentons une revue de littérature qui examine l'état-de-l'art, les différentes stratégies utilisées dans la création d'agents conversationnels ainsi que leurs différentes classifications.

Une attention particulière est portée sur les limites des approches existantes. Ainsi, une comparaison des approches génératives les plus récentes est faite avec la mise en évidence de leurs faiblesses dans le domaine du TAL, en particulier les tâches de traduction automatique et de génération de dialogue multi-tours.

Nous terminons le chapitre par une analyse critique globale et une introduction préliminaire des solutions que nous proposons.

2.1 Introduction

Les agents conversationnels (AC), appelé aussi chatbots, existent depuis presque aussi longtemps que les ordinateurs. L'idée d'une intelligence artificielle capable de dialoguer avec un humain est un idéal qui a été recherché depuis qu'Alan Turing a proposé le « jeu d'imitation » pour tester l'intelligence artificielle (Turing, 1950). Ce jeu d'imitation a fini par être surnommé, dans la langue vernaculaire commune, le test de Turing (Schuetzler, 2015).

L'idée derrière le test de Turing était de répondre à la question « Les machines peuvent-elles penser ? » (Turing, 1950). Afin de tester la capacité de réflexion d'une machine, Turing a proposé un test dans lequel un juge humain serait placé dans une conversation médiatisée entre un humain et une machine et serait chargé de déterminer lequel des deux est la machine. Aujourd'hui, ce test est généralement déployé dans un message instantané ou une conversation par clavardage dans un environnement assisté par ordinateur, comme dans le concours annuel du prix Loebner (Bradeško et Mladenić, 2012; Schuetzler, 2015).

Avant l'arrivée des techniques d'apprentissage automatique, le développement des agents conversationnels impliquait principalement la spécification des règles avec lesquelles la réponse serait générée. Ces règles peuvent aller de la simple correspondance de modèle à la structure grammaticale de la phrase qui peut être utilisée pour comprendre le contexte de la conversation. L'approche classique de construction d'agents conversationnels utilise généralement des solutions basées sur un métalangage tel que AIML [1] avec une analyse de NLP [2]/NLU [3].

2.2 Classification des agents conversationnels

Les agents conversationnels peuvent être classés en fonction de plusieurs critères (Figure 2.2). Ces critères peuvent inclure la philosophie de conception de ces agents, ou la mesure dans laquelle le contexte doit être utilisé et pris en compte pour comprendre la conversation tout comme, le type et le but de la conversation pour lequel l'agent doit être conçu (Cahn, 2017).

---

1. AIML :Artificial Intelligence Markup Language
2. NLP :Natural Language Processing
3. NLU : Natural Language Understanding

2.2.1 Conversations multi-tours vs à un tour

À mesure que la durée de la conversation augmente, il devient plus difficile de l'automatiser. Basé sur la longueur de la conversation, nous avons deux types de conversations. D'abord, il y a les conversations en texte à un tour où une seule réponse est produite. Par exemple, donner une réponse appropriée lorsqu'une question spécifique est posée par l'utilisateur. D'un autre côté, il y a les conversations longues ou très longues (multi-tours ou multi-contextes) où il y a beaucoup d'informations échangées dans le passé qui devraient être conservées pour en tirer un bon résultat (Li *et al.*, 2016b).

La conversation multi-tours correspond au type de conversation traité dans cette thèse.

2.2.2 Domaine ouvert ou domaine fermé

Lorsque les humains interagissent, ils peuvent lancer une discussion dans un domaine et passer ensuite à un autre. De telles conversations sont connues comme des conver-

sations de domaine ouvert. Dans ces conversations, le domaine peut changer au fil du temps. Ces modèles ne sont pas conçus pour servir un objectif spécifique. Un exemple d'une telle conversation est l'interaction dans les sites de médias sociaux comme Facebook, Twitter, etc. (Zhou et Wang, 2018). En raison de la quantité importante de connaissances requises pour générer une réponse raisonnable, il est assez difficile d'imiter de telles conversations (Li *et al.*, 2016b).

Dans une configuration de domaine fermé, une connaissance limitée spécifique au domaine est nécessaire pour générer une réponse appropriée à l'entrée. Un exemple d'une telle catégorie est un système de service à la clientèle. Dans de tels systèmes, il n'y a pas d'écart par rapport au but spécifique pour lequel le système est conçu. À cette fin, la conversation se concentre principalement sur la spécificité du domaine et la génération de la réponse le plus rapidement possible. Ces systèmes sont généralement formés en obtenant des données spécifiques au domaine, afin de générer des réponses en fonction du contexte (Li *et al.*, 2016b).

Le domaine ouvert correspond au type du domaine traité dans cette thèse.

2.2.3 L'architecture

La classification basée sur l'architecture permet de classifier les agents selon les techniques d'apprentissage utilisées. Les agents basés sur la récupération utilisent un ensemble de réponses prédéfinies, puis appliquent une sorte d'heuristique afin de choisir une réponse appropriée supportant à la fois l'entrée et le contexte. Une variété d'heuristiques peut être appliquée afin de choisir une réponse appropriée. Elle peut être basée sur le concept assez simple d'une correspondance d'expressions basée sur des règles, ou peut être aussi complexe que l'utilisation d'une combinaison de techniques d'apprentissage automatique. Les systèmes basés sur la récupération ne produisent pas de nouvelles réponses, mais en choisissent simplement une parmi un ensemble de réponses

prédéfinies (Cuayáhuitl *et al.*, 2019).

Cependant, les modèles génératifs surmontent cette dépendance aux réponses prédéfinies en générant de nouvelles réponses qui sont construites en appliquant un ensemble de techniques. Ces modèles sont généralement basés sur diverses approches de traduction automatique. Néanmoins, dans ces modèles, nous traduisons une entrée en réponse, au lieu de traduire un texte dans une langue source vers une langue cible. Les méthodes mentionnées ci-dessus ont leurs propres avantages et inconvénients. Les approches basées sur la récupération ne font pas d'erreurs grammaticales parce qu'elles choisissent simplement une réponse à partir d'un ensemble de réponses prédéfinies. En revanche, elles ne sont pas capables de gérer des conditions inconnues pour lesquelles il n'existe pas de réponse prédéfinie appropriée dans le référentiel de réponses. Pour les mêmes raisons, ces modèles sont incapables d'avoir une compréhension du contexte précédent (Sankar, 2020). Les informations telles que les noms de lieu, de personnes ou toute autre chose mentionnée précédemment dans la conversation ne peuvent pas être évoquées dans ces modèles. D'un autre côté, les modèles génératifs possèdent la capacité de se référer à des informations passées. Cela rend l'interaction homme-machine encore plus humaine. Ces modèles sont toutefois très difficiles à entraîner et ont besoin de grands corpus (Cuayáhuitl *et al.*, 2019; Sankar, 2020).

Les résultats des modèles basés sur la récupération et les modèles génératifs peuvent être améliorés en utilisant des techniques d'apprentissage profond (Cuayáhuitl *et al.*, 2019). La tendance actuelle de la technologie tend vers les méthodes génératives. Il existe actuellement des architectures d'apprentissage profond telles que celles basées sur les modèles séquence vers séquence, le Transformer et les architectures hiérarchiques récurrentes qui fournissent divers mécanismes pour générer du texte (Surmenok, 2016; Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Park *et al.*, 2018; Xing *et al.*, 2018).

2.3 Architecture basée sur les règles

Les modèles de substitution et la recherche de motifs représentent la méthodologie la plus utilisée par les agents conversationnels. Bien que la complexité de ces algorithmes puisse différer, l'idée fondamentale qui les sous-tend reste la même. Des règles de modèle simples et basiques ont été utilisées dans les premiers agents comme ELIZA (Weizenbaum, 1966) et PARRY (Colby *et al.*, 1972).

Dans cette approche, la structure de la phrase est identifiée et une réponse prédéfinie est modifiée en fonction des variables caractéristiques de la phrase (Jurafsky et Martin, 2017, Chapitre 29). Fondamentalement, les conversations passées sont faites sous une forme généralisée.

Cette méthodologie est couramment utilisée dans les robots question-réponse. L'inconvénient de cette approche est que les réponses sont assez prévisibles, répétitives et manquent de contact humain. En outre, il n'y a généralement pas de stockage des réponses passées, ce qui peut conduire à des conversations en boucle. À mesure que les algorithmes d'appariement de motifs deviennent plus complexes, les réponses deviennent très limitées, ce qui peut mener à des conversations sans intérêt (Meffert, 2006).

Un pas en avant important pour les approche basées sur les règles est venu avec la création d'A.L.I.C.E. (Artificial Linguistic Internet Computer Entity). La principale contribution d'A.L.I.C.E. est l'introduction du métalangage AIML (Artificial Intelligence Markup Language). AIML est un langage dérivé de XML utilisé pour gérer la connaissance qui permet la création des agents de conversation en utilisant la correspondance de modèle (Motifs) pour analyser l'entrée de l'utilisateur. De nouvelles variantes d'AIML offrant les mêmes fonctionnalités ont vu le jour ensuite, comme MPML (Multimodal Presentation Markup Language) (Mori *et al.*, 2003).

Une autre variante de ces langages est proposée dans ChatScript qui vise à être un suc-

cesseur du langage AIML. Bien qu'AIML soit simple et facile à apprendre, il possède des algorithmes de correspondance de motifs relativement faibles et est difficile à maintenir. Contrairement à AIML qui est directement basé sur la correspondance de modèle (Motif), ChatScript est composé de règles associées à des sujets. Il trouve d'abord le meilleur sujet qui correspond à la chaîne de la requête de l'utilisateur et exécute une règle qui correspond le mieux à ce sujet. Suzette de Brian Wilcox est le premier agent conversationnel basé sur ChatScript (Wilcox *et al.*, 2010). ChatScript introduit des « concepts » qui sont des ensembles de mots similaires par rapport à l'une ou l'autre signification ou à toute autre propriété comme les parties du discours. Un concept de tous les noms ou adverbes peut être créé. Il existe également une base de données de concepts (les concepts sont modulaires) et par conséquent, des concepts déjà définis peuvent être utilisés. Dans le cas d'AIML, des catégories séparées doivent être créées pour chacun de ces mots. ChatScript inclut également de la mémoire à long terme sous la forme de variables qui peuvent être utilisées pour stocker durant la conversation des informations spécifiques comme le nom, le prénom ou une date de l'utilisateur. Ces variables peuvent être réutilisées comme historique afin de générer les réponses (McNeal et Newyear, 2013).

2.4 Architecture basée sur l'apprentissage automatique

Les agents basés sur l'apprentissage automatique utilisent des conversations humaine-humaine longues à la place de règles construites à la main, ou dans certains cas des réponses à des questions humaines apprises à partir d'un corpus. Gao *et al.* (2018b) résument certains des corpus disponibles, tels que les conversations sur les plateformes de discussion, par exemple Twitter (Zhou et Wang, 2018), ou dans les dialogues de sous-titrage des films, qui sont disponibles en grande quantité et qui ressemblent à une conversation naturelle (Forchini, 2013). Les réponses peuvent même être extraites de phrases dans des corpus de textes qui ne sont pas des dialogues.

Il existe deux types de systèmes qui sont basés sur les corpus :

- ⋆ Les systèmes basés sur la recherche d'information (ou à base de récupération).
- ⋆ Les systèmes basés sur l'approche générative.

2.4.1 Architecture basée sur la recherche d'informations

Le principe derrière les agents basés sur la recherche d'informations (RI) est de répondre au tour[4] X d'un utilisateur en repérant un énoncé Y approprié à partir d'un corpus de texte (Wical, 2002). Les différences entre ces systèmes se trouvent dans leurs choix du corpus et comment ces derniers décident ce qui compte dans les requêtes. Un choix commun de corpus est de collecter des bases de données de conversations humaines. Celles-ci peuvent provenir de plateformes de microblogging comme Twitter. Une autre approche consiste à utiliser des corpus de dialogues de films. Une fois qu'un agent de conversation est mis en pratique, les tours que les humains utilisent pour répondre à l'agent peuvent être utilisés comme données conversationnelles supplémentaires pour l'entraînement Gao *et al.* (2018b). Étant donné le corpus et la phrase de l'utilisateur, les systèmes RI peuvent utiliser n'importe quel algorithme de récupération pour choisir une réponse appropriée du corpus. La méthode la plus simple est d'envoyer la réponse au tour t le plus similaire à la requête de l'utilisateur q qui existe dans un corpus C (On utilise par exemple le cosinus de la similarité avec q afin de retourner le tour suivant, c'est-à-dire la réponse humaine à t dans le corpus C selon l'équation 2.1 (Jurafsky et Martin, 2017, Chapitre 29)).

$$\hat{r} = \text{réponse}\left(argmax_{t \in C} \frac{q^T t}{\|q\| \, \|t\|}\right) \tag{2.1}$$

L'idée est que nous devrions rechercher un tour qui ressemble le plus au tour de l'utilisateur, et retourner la réponse humaine à ce tour (Jafarpour et Burges, 2010; Leuski et

4. un tour est le rôle d'un interlocuteur dans le dialogue

Traum, 2011).

Dans chaque cas, n'importe quelle fonction de similarité peut être utilisée. Le plus souvent, des cosinus sont calculés sur des mots (en utilisant des méthodes de pondération telles que *TF-IDF*).

Bien que retourner la réponse au tour le plus semblable est un algorithme plus intuitif, retourner l'énoncé le plus similaire semble mieux fonctionner en pratique, peut-être parce que la sélection de la réponse ajoute une autre couche d'indirection qui peut maintenir des mots partagés durant la conversation (Ritter *et al.*, 2011; Wang *et al.*, 2013). L'approche basée sur la RI peut être étendue en utilisant plus de fonctionnalités que les mots de la requête (comme les mots dans la discussion antérieure, ou des informations sur l'utilisateur). Cleverbot (Carpenter, 2017) et Microsoft Xiaoice (Lee *et al.*, 2017) sont des exemples d'applications commerciales de cette approche. Au lieu de simplement utiliser des corpus de conversation, l'approche basée sur la RI peut étendre la recherche des réponses à partir de corpus (non-dialogue). Par exemple, les agents qui veulent générer des tours informatifs, tels que les réponses aux questions, peuvent chercher l'information dans Wikipédia ou dans des sites Web spécifiés, afin d'extraire des phrases manquantes du corpus (Yan *et al.*, 2016).

2.4.2 Architecture basée sur l'approche générative

Une façon d'utiliser un corpus pour générer un dialogue est de considérer la génération de réponses comme une tâche de traduction. Cette dernière tâche est appliquée entre la requête de l'utilisateur et la réponse du système. Typiquement, deux principaux types de systèmes pilotés par les données sont discernés :

- ⋆ La méthode statistique est basée sur la traduction automatique statistique (SMT [5])

5. SMT : Statistical machine translation

pour exploiter les modèles de haute fréquence à l'aide d'une combinaison de paires de phrases alignées qui se contraignent par le chevauchement lexical (par exemple, (Ritter *et al.*, 2011)).

⋆ La méthode de génération neuronale consiste à produire une réponse la plus probable en sachant un contexte conversationnel. Ensuite, on échantillonne la distribution de probabilité mot à mot de ce modèle (Vinyals et Le, 2015).

Cependant, il est vite devenu clair que la tâche de génération de réponses était trop différente de la traduction automatique. Au coeur de la traduction automatique, les mots ou les expressions dans les phrases source et cible ont tendance à bien s'aligner les uns avec les autres ; mais un énoncé d'utilisateur peut ne partager aucun mot ou phrase avec une réponse cohérente. Simultanément, Shang *et al.* (2016); Sordoni *et al.* (2015b) ont proposé des modèles pour la génération de réponses basés sur le modèle séquence à séquence, après son succès dans la traduction automatique (Seq2Seq (Vinyals et Le, 2015)).

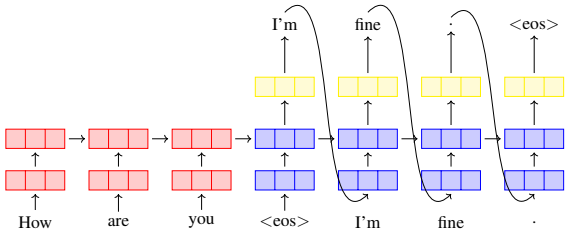


Figure 2.1: Le modèle Seq2Seq pour la génération de la réponse dans le dialogue.

Un certain nombre de modifications sont nécessaires au modèle de base Seq2Seq pour l'adapter à la tâche de génération de réponses. Par exemple, ce modèle a tendance à produire des réponses prévisibles, mais répétitives et donc ennuyeuses comme « I'm ok ok » ou « I don't know », et qui peuvent mettre fin à la conversation (Li *et al.*, 2016a). Un

autre problème avec cette architecture est son incapacité à modéliser le contexte antérieur au fur et à mesure qu'on avance dans une conversation longue. Bien que ce modèle n'utilise qu'un seul tour de conversation. Cela peut-être fait en permettant au modèle de voir les changements antérieurs de dialogue, comme celui qui a été utilisé par le modèle hiérarchique. On peut aussi résumer les informations sur plusieurs tours précédents (Lowe *et al.*, 2017). Enfin, les générateurs de réponses Seq2Seq se concentrent sur la génération de réponses uniques et n'ont donc pas tendance à faire du bon travail. Cela est dû au fait que ces mêmes générateurs de réponses ne sont pas en mesure de générer de façon continue des réponses qui coïncident avec plusieurs changements. Ce type de problèmes peut être résolu en utilisant l'apprentissage par renforcement, ainsi que des techniques comme les réseaux antagonistes génératifs (GAN [6]), pour apprendre à choisir des réponses qui rendent la conversation globale plus naturelle (Li *et al.*, 2016c, 2017a).

En 2017, Vaswani *et al.* (2017) ont proposé le modèle Transformer. Ce modèle est caractérisé par le remplacement des RNR par un mécanisme basé sur l'auto-attention. De nombreux modèles essaient d'entraîner le Transformer pour la génération de dialogue court, comme DialoGPT (Zhang *et al.*, 2020c) et DodecaDialogue (Shuster *et al.*, 2019).

Bien que ce qui précède soit basé sur une architecture Seq2Seq, il a eu du succès avec les tâches de génération d'un tour, comme les tâches Question/Réponse. Cependant, les chercheurs ont rapidement tourné leurs dos à ce modèle en faveur des modèles génératifs hiérarchiques pour les tâches de dialogue multi-tours, comme nous pouvons le voir dans la section 2.7.

Le tableau 2.1 montre quelques exemples de réponses générées par un modèle Seq2Seq simple d'une part, et un modèle Seq2Seq formé par un réseau antagoniste (GAN)

6. GAN : Generative Adversarial Network

d'autre part, pour produire des réponses à un tour (Li *et al.*, 2017a).

Interlocuteur	énoncé
Entrée	tell me ... how long have you had this falling sickness ?
Simple Seq2Seq	i'm not a doctor.
Antagoniste	a few months, i guess .
Entrée	so i had the doctors test sammy 's response to conditioning .
Simple Seq2Seq	sammy wrote the test sammy wrote the test .
Antagoniste	so he took the pills .
Entrée	they didn 't have much success with this problem commander .
Simple Seq2Seq	they 're not the only ones who have been in the system .
Antagoniste	can we find someone else ?

Tableau 2.1: Quelques exemples de réponses générées par un modèle Seq2Seq simple et par un réseau antagoniste (Li *et al.*, 2017a).

2.5 Agent conversationnel hybride

Récemment, des approches hybrides ont été proposées, comme *AliMeChat* (Qiu *et al.*, 2017) qui intègre à la fois les modèles RI et l'approche générative à travers un modèle Seq2Seq. Dans ce système, on utilise un modèle de reclassement de Seq2Seq attentif pour optimiser les résultats communs. Plus précisément, pour une question, on utilise d'abord un modèle RI pour extraire un ensemble de paires Question/Réponse afin de les utiliser comme des réponses candidates. Ensuite, ce système relance les réponses candidates par un modèle Seq2Seq attentif. Dans le cas où le meilleur candidat a un score supérieur à un certain seuil, il sera une réponse au dialogue ; dans le cas contraire, la réponse sera fournie par un modèle basé sur la génération.

Cuayáhuitl *et al.* (2019) ont proposé de leur côté un autre modèle hybride, *MilaBot*, en utilisant l'apprentissage par renforcement. Le système consiste en un ensemble de modèles de génération et de récupération du langage naturel. Son architecture utilise à la fois des modèles de sac de mots adoptant une approche RI, un réseau de neurones

Seq2Seq et un modèle neuronal hiérarchique doté d'une variable latente (VHRED). Ensuite, ce système applique l'apprentissage par renforcement à des données de l'annotation manuelle de *Crowdsourcing* afin d'apprendre la bonne réponse choisie par les annotateurs. Le système a été formé pour sélectionner une réponse appropriée à partir des modèles de son ensemble.

2.6 Architecture basée sur les schémas (Frame-based dialog agents)

Les systèmes de dialogue orientés vers les tâches modernes sont basés sur une ontologie de domaine. Cette dernière est aussi une structure de connaissances représentant les types d'intentions des utilisateurs ainsi que les entités du domaine. Le système a la capacité d'extraire l'intention portée par un concept ontologique et l'associer avec le dialogue de l'utilisateur (Jurafsky et Martin, 2017, Chapitre 3). Dans ce cas, l'ontologie définit un ou plusieurs champs. Chaque champ est défini par un type de valeurs prises du dialogue et mises dans un tableau nommé *slot*. Cette architecture à base de schémas a été introduite dans le système *GUS* pour la planification des voyages (Bobrow *et al.*, 1977) et s'est propagée à la plupart des assistants numériques commerciaux modernes (Jurafsky et Martin, 2017, Chapitre 3). Le tableau 2.2 montre une conversation avec le système *GUS* original.

L'ensemble des emplacements dans un slot de style *GUS* spécifie ce que le système doit savoir, et le remplissage de chaque emplacement est contraint aux valeurs d'un type sémantique particulier, comme indiqué dans le tableau 2.3. Par exemple, un événement peut être de type *ville* (donc prendre des valeurs comme Montréal ou Alger) ou de type date et heure, compagnie aérienne. Les agents de dialogue modernes, basés sur des slots, peuvent avoir une structure hiérarchique. Ainsi, par exemple le type date est lui-même un slot avec un ensemble de types comme des entiers ou des membres d'ensembles tels que les noms des jours de la semaine (Jurafsky et Martin, 2017, Chapitre 3).

GUS :	Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go ?
Client :	I want to go to San Diego on May 28
GUS :	What time do you want to leave ?
Client :	I must be in San Diego before 10 am
GUS :	Would you like P.S.A. flight 102 that arrives at 9.15 am ?
Client :	What is the next flight ?
GUS :	Air California flight 310 that leaves at 8.30am and arrives at 10.00am
Client :	I'll take the first one
GUS :	In what name should I make a reservation, please ?
Client :	Dan Bobrow
GUS :	I have confirmed the following flight :
Client :	P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am.
GUS :	What date do you want to return on ?
Client :	On Friday in the evening.
GUS :	Would you like the flight that leaves at 7.45 pm ?
Client :	That's fine.
GUS :	I have confirmed the following flight : P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm Thank you for calling. Goodbye

Tableau 2.2: Interaction avec le système *GUS* dans le domaine du voyage (Bobrow *et al.*, 1977; Jurafsky et Martin, 2017).

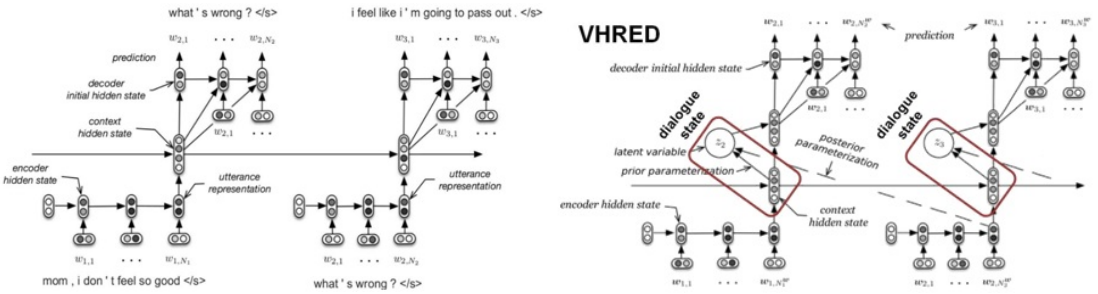
Champs	Type
DÉPART VILLE	ville
DESTINATION VILLE	ville
DÉPART HEURE	heure
DÉPART DATE	heure
ARRIVÉE HEURE	heure
ARRIVÉE DATE	date

Tableau 2.3: Les types dans un Slot représenté par le système *GUS*

Le but de cette décomposition du dialogue en des composantes élémentaires est la compréhension du langage naturel et l'extraction de trois éléments de l'énoncé de l'utilisateur. La première information est la classification des domaines : cet utilisateur parle-t-il par exemple des compagnies aériennes, de la programmation d'un réveil ou du traitement de son calendrier ? Bien sûr, cette tâche de classification n'est utile que pour les systèmes à domaine unique qui sont axés sur le but, sachant que les systèmes de dialogue à plusieurs domaines sont la norme actuelle. Cependant, l'utilisation de cette

approche rend le système très complexe. Le second élément est la détermination de l'intention de l'utilisateur. Selon la question, quelle tâche générale ou quel objectif l'utilisateur tente-t-il d'accomplir ? Par exemple, la tâche peut être de rechercher un article ou d'afficher un vol ou de supprimer un rendez-vous du calendrier (Woudenberg, 2014; Jurafsky et Martin, 2017). Le troisième élément est le remplissage des slots qui aide à l'exécution des tâches (par exemple, consulter des données dans une base de données) (Bobrow *et al.*, 1977; Jurafsky et Martin, 2017).

2.7 Génération de réponses multi-tours



(a) Hierarchical Recurrent Encoder Decoder (Sordoni *et al.*, 2015a)

(b) Variable latent Hierarchical Recurrent Encoder Decoder (Serban *et al.*, 2017c)

Figure 2.2: Architectures HRED (a) vs VHRED (b)

Avant, les architectures des systèmes de dialogues classiques créaient un gestionnaire qui contrôlait l'historique des conversations et le raisonnement derrière les requêtes. Ce raisonnement est extrait par l'intermédiaire d'un annotateur de l'intention de l'utilisateur. Compte tenu de l'absence d'un gestionnaire de dialogue dans les modèles génératifs de *Seq2Seq* dédiés à une conversation à un seul tour ; la génération de réponses multi-tours représente un grand défi (Wang et Yuan, 2016).

Il est clair qu'un agent de conversation multi-tours n'utilise pas seulement la dernière phrase pour répondre à une requête, mais il maintient des représentations internes en

mémoire afin de garder la cohérence de dialogues. Les mémoires internes peuvent être explicites ou implicites. Elle sont validées par certaines tâches externes telles que la validation des réponses aux questions (Wang et Yuan, 2016).

À côté de la génération de séquences normales, des mécanismes différents doivent être explorés en s'inspirant de la façon dont les humains utilisent la conversation longue (multi-tour). Motivés par le fait que les modèles Seq2Seq ne parviennent pas à capturer le flux hiérarchique pour les séquences de phrases, Sordoni *et al.* (2015a) ont proposé une approche basée sur une structure hiérarchique dite HRED [7]. Leur but principal est la modélisation conjointe des requêtes web (composés de jetons) et de la structure interactive (comprenant des énoncés) (Figure 2.2a). À cet effet, Serban *et al.* (2016a) ont utilisé le modèle hiérarchique (HRED) pour la génération de dialogues multi-tours. Leurs motivations justifiées par la structure temporelle à long terme du dialogue, ont été prises en considération par le modèle proposé qui simule un suivi de l'état du dialogue et ainsi permet une conversation à plusieurs itérations.

Une autre approche complémentaire proposée vise à généraliser l'architecture Seq2Seq à des entrées multiples et des paires de sorties où chaque séquence représente son propre processus stochastique. Motivés par le fait que les modèles existants ont été incapables de générer des réponses significatives en prenant en compte le contexte de dialogue, Serban *et al.* (2017a) ont proposé les réseaux de neurones multi-résolution capables de modéliser des séquences de sorties complexes avec des dépendances à long terme. Plus précisément, cette architecture double hiérarchique permet la modélisation de multiples séquences en parallèle, l'une étant une simple séquence de jetons du langage naturel et l'autre étant une séquence représentant des jetons secondaires de haut niveau (inférence), de telle sorte que la représentation des noms suggérés ou activités-entités serait possible (Serban *et al.*, 2017a). L'inconvénient majeur pour ce modèle est l'utilisation

7. HRED : Hierarchical Recurrent Encoder-Decoder

d'une approche supervisée, car le modèle utilise des corpus d'entraînement étiquetés.

Dans le but de gérer les conversations longues ouvertes à composition non limitées, les systèmes de dialogue de bout en bout devraient suivre efficacement de nombreuses variables telles que le sujet de la conversation, le style d'interaction, le sentiment, etc. ; qui sont latentes par nature, c'est à dire inférables à partir de l'observation. Dans une conversation naturelle inter-humaine, ces variables influencent le dialogue et les gens ajustent leurs réponses en conséquence. Un système de dialogue de bout en bout devrait également répondre à ces caractéristiques de génération de conditionnement en fonction de ces variables latentes et ainsi devrait prédire comment l'étape en vigueur influence la prochaine étape générée.

Serban *et al.* (2017c) ont proposé de représenter cette variabilité, l'incertitude et l'ambiguïté dans les états actuels et à venir, en augmentant l'architecture HRED [8] (Figure 4.2) avec des variables stochastiques latentes qui couvrent éventuellement un nombre variable d'évènements externe influençant la conversation. Le processus de génération hiérarchique est représenté par une distribution probabiliste conditionnée par des variables latentes qui permettent une modélisation complexe.

Cependant, le problème lié à la combinaison du VAE [9] avec les décodeurs RNR est l'effet de dégénération de la variable latente par le modèle durant la conversation (Bowman *et al.*, 2016; Park *et al.*, 2018), ce qui empêche le décodeur d'apprendre la dépendance pouvant exister entre la variable latente et les données, et de la réduire à un simple RNR. Pour surmonter ce problème, Park *et al.* (2018) propose les RNR de conversation hiérarchique variationnelle (VHCR [10]) qui utilisent une structure hiérarchique de

8. HRED :Hierarchical Recurrent Encoder Decoder

9. VAE : Variational Autoencoder

10. VHCR : Variational Hierarchical Conversation RNN

variables latentes et la régularisation de la participation des énoncés dans la conversation (*utterance drop regularization*). La solution proposée réduit l'impact du problème, mais ne l'élimine pas.

Wang et Jiang (2019) ont proposé une simplification du HRED, appelée *SHRED* [11]. Les auteurs proposent de simplifier les couches supérieures de HRED en utilisant une variante GRU appelée *Scalar Gated Unit (SGU)* pour l'encodeur de niveau d'énoncé. Toutefois, l'optimisation proposée n'a aucun effet sur le résultat produit par le modèle d'origine.

Par ailleurs, Tian *et al.* (2017) ont proposé une amélioration supplémentaire pour le modèle hiérarchique fondé sur HRED, en ajoutant un mécanisme d'attention entre le résultat de décodeur et le contexte généré par l'encodeur des énoncés. Ce modèle est nommé *Weighted vectors Sequence*, où un mécanisme d'attention analyse les états cachés de l'encodeur pour établir des liens à court terme entre les vecteurs d'entrée et les appliquer à la sortie. Ce modèle utilise une architecture hiérarchique et hérite toutes les limites des RNRs.

Des améliorations ont été apportées au modèle précédent, en utilisant les RNR bidirectionnel avec une attention supplémentaire pour l'encodeur de mots (Xing *et al.* (2018)). Ce modèle nommé *Hierarchical Recurrent Attention Network (HRAN)*, jouit d'un mécanisme de double attention qui analyse les états cachés du premier et du deuxième encodeur afin d'établir des liens à court terme entre les mots d'entrée pour les appliquer à la sortie. Ce modèle utilise une architecture hiérarchique et hérite de toutes les limites des RNRs.

Plus récemment, Zhang *et al.* (2019a) ont proposé un autre modèle inspiré de HRED

11. SHRED : Simplified HRED

et Transformer (Vaswani *et al.*, 2017), appelé ReCoSa [12] (contexte pertinent avec auto-attention), basé sur une augmentation du modèle avec l'auto-attention. ReCoSa utilise un encodeur de type RNR (LSTM) au niveau du mot, afin d'obtenir la représentation initiale de chaque contexte d'énoncé. Ensuite, le mécanisme d'auto-attention est utilisé pour mettre à jour à la fois le contexte et la représentation de réponse masquée. Les poids d'attention entre chaque contexte et les représentations de réponse sont calculés et utilisés dans le processus de décodage qui suit.

L'avantage de ce modèle est la réduction du nombre de RNRs utilisés. En revanche, il n'élimine pas définitivement le problème, car ReCoSa inclut les RNRs dans la phase de départ de l'encodage des mots afin de définir un contexte pour chaque énoncé. De plus, deux encodeurs sont utilisés : le premier est un RNR permettant l'apprentissage de la composition des mots de chaque énoncé et le second utilise l'auto-attention pour apprendre la composition des énoncés en conversation.

Un autre modèle capture les informations sur le sujet de chaque contexte durant la conversation afin de l'utiliser comme facteurs additionnels dans le processus de décodage. Zhang *et al.* (2020b) ont proposé un modèle, nommé STAR-BTM (Biterm Topic Model). Ce modèle est composé de deux modules, le premier est un module de détection de sujet nommé Biterm afin de détecter les sujets, le deuxième est un modèle HRED. Les poids d'attention au niveau du sujet sont calculés par Biterm en fonction de la représentation du sujet de chaque conversation. Ensuite, les poids d'attention et la distribution des sujets sont utilisés dans le processus de décodage pour générer les réponses correspondantes. Ce modèle ne propose aucune amélioration au modèle original de HRED. C'est à dire, le module Biterm peut être ajouté dans toutes les architectures hiérarchiques.

Olabiyi *et al.* (2018) ont proposé une approche d'apprentissage basée sur les réseaux

12. ReCoSa : Relevant Contexts with Self-Attention

antagonistes génératifs (GAN [13]) pour générer des réponses de dialogue multi-tours. Le générateur du GAN est un réseau encodeur-décodeur récurrent hiérarchique (HRED) et le discriminateur est un RNR bidirectionnel au niveau du mot qui partage les encapsulations de texte et de mots avec le générateur. L'approche perturbe l'espace latent du générateur par un bruit Gaussian afin de générer plusieurs réponses possibles. Ce modèle ne propose aucun changement architectural au modèle de HRED. C'est à dire, la technique des réseaux antagonistes génératifs peut être appliquée à toutes les architectures génératives.

Bonetta *et al.* (2021) présentent un modèle basé sur le transformer pour la génération de réponse de dialogue multi-tours. L'approche hybride proposée combine un modèle génératif basé sur le Transformer et un mécanisme à base de récupération (approche basée sur la recherche d'information) qui exploite les informations mémorisées dans les données d'apprentissage via l'algorithme de la recherche de k-plus proche voisin. Ce modèle ne propose aucun changement architectural au modèle originel du Transformer. De plus, l'augmentation du modèle par un mécanisme a base de récupération est applicable à toutes les architectures génératives.

Tan *et al.* (2019) présentent un modèle basé sur le Transformer pour la génération de réponse de dialogues multi-tours. L'approche combine deux encodeurs de Transformer. Le premier utilisant un contexte pour l'historique des énoncés précédents et le second utilisant les mots des énoncés. Ce modèle ressemble à celui de HRED et ReCoSa vus précédemment, sauf qu'il n'utilise pas de RNR.

En 2021, Wang *et al.* (2021) proposent CDial-GPT, un modèle de dialogue chinois basé sur l'apprentissage par transfert en utilisant le modèle de langue GPT. Ce modèle utilise un filtrage sur les données d'apprentissage à l'aide de deux classificateurs. Le premier vise à garder les phrases moins bruyantes mais complètes, alors que Le second vise

13. GAN : Generative Adversarial Networks

l'exclusion des conversations dépendantes de contextes externes en dehors du texte, ce qui les rend difficiles à comprendre. L'inconvénient majeur de ce modèle est sa dépendance à la langue chinoise utilisée.

Zhou *et al.* (2021) proposent un Transformer pré-entraîné nommé EVA. L'encodeur et le décodeur de ce modèle sont entraînés sur un ensemble de données qui contient 1,4 milliard de paires contexte-réponse utilisées comme corpus de préformation d'EVA. Les mêmes auteurs proposent EVA2.0 en utilisant le modèle de dialogue EVA avec 2,8 milliards de paramètres, et ils adoptent une architecture basée sur le Transformer combinée à un encodeur bidirectionnel et un décodeur unidirectionnel pour la modélisation du dialogue (Gu *et al.*, 2022).

Mi *et al.* (2022) proposent l'agent PANGUBOT préformé sur un modèle GPT. Dans ce travail, les auteurs ont montré que sur la base d'un grand modèle de langage bien formé comme GPT, sa formation sur un ensemble de données plus petit peut atteindre de bonnes performances de dialogue.

Les modèles EVA, EVA2 et PENGUBOT sont conçus pour l'apprentissage par transfert en utilisant exclusivement les corpus de langue chinoise.

Malgré les progrès réalisés au sein de tous ces travaux, la génération des réponses multi-tours reste très loin des attentes.

Nous avons remarqué qu'il est difficile de retracer les règles de causalité entre les mots des énoncés et les mots de la réponse dans les modèles hiérarchiques. Ceci est dû à la présence d'un encodeur intermédiaire pour les énoncés entre l'encodeur des mots sources et le décodeur des mots cibles. Ainsi, ces modèles ont des problèmes avec les tâches de traduction du séquences comme la traduction automatique. Nous avons aussi observé que le point commun de tous ces modèles est la formule de l'équation 2.2 que nous devrions utiliser par la suite (Belainine *et al.*, 2022a, 2020c) comme point de

départ dans nos recherches, telle que proposée par Sordoni *et al.* (2015b).

$$P(E_m|E_1,\cdots,E_{m-1}) = \prod_{n=1}^{N_m} P(w_n| \overbrace{w_1,\cdots,w_{n-1}}^{\text{anciens mots dans l'énoncé}}, \overbrace{E_1,\cdots,E_{m-1}}^{\text{ancienne énoncés}}) \tag{2.2}$$

Une séquence de dialogues à m tours est représentée en tant que $d = (E_1, E_2, \cdots, E_i, \cdots, E_{m-1})$. L'encodeur prend tous les éléments jusqu'à E_{m-1} comme entrées et le décodeur essaie de prédire $E_m = (w_1, \cdots, w_n)$ mot par mot (Belainine *et al.*, 2022a, 2020c).

2.8 Corpus

Lorsque nous envisageons d'appliquer l'apprentissage automatique à n'importe quelle tâche, l'une des premières choses à faire est de chercher l'ensemble de données dont nous aurions besoin pour former un modèle. Pour les modèles de dialogue, nous avons besoin d'un grand jeu de données de conversations, appelé aussi corpus de dialogues (Gao *et al.*, 2018b). Ces corpus contiennent généralement des interactions dialoguées, même si parfois plus de deux interlocuteurs peuvent être impliqués. Ces derniers sont particulièrement intéressants pour la recherche et l'apprentissage des agents conversationnels. On distingue deux types de corpus (Gao *et al.*, 2018b) :

1. Corpus de dialogue général
2. Corpus de dialogue axé sur les buts

2.8.1 Corpus dialogue général

Dans ce type de dialogue, on trouve des sujets de conversation appartenant à des domaines différents.

Le corpus de dialogue *Ubuntu* (Lowe *et al.*, 2017) est utilisé pour évaluer plusieurs agents neuronaux. C'est un ensemble de données contenant près d'un million de dialogues multi-tours, avec un total de plus de 7 millions d'énoncés et de 100 millions de mots. Cela fournit une ressource unique pour former des modèles de langues à base de

réseaux de neurones, pouvant utiliser de grandes quantités de données d'entraînement. L'ensemble de données ne possède pas seulement la propriété multi-tours des conversations, mais aussi le suivi des états de dialogue (corpus de dialogue long) (Lowe *et al.*, 2017).

Le corpus des dialogues de films *Movie-DIC* (Danescu-Niculescu-Mizil et Lee, 2011) est un autre type que nous rencontrons souvent. Ce corpus contient une vaste collection de conversations fictives extraites de scripts de films bruts riches en métadonnées. Ce corpus a été extrait de la collection OpenSubtitles (Müller et Volk, 2013) et contient 220k échanges de conversations entre 10 et 292 paires de personnages de films (Danescu-Niculescu-Mizil et Lee, 2011).

Le jeu de données de dialogue *DailyDialog* contient 13 118 dialogues de bonne qualité. Les dialogues de l'ensemble de données reflètent notre façon de communiquer quotidiennement et couvrent différents sujets dans plusieurs domaines. L'ensemble de données a été étiqueté manuellement avec l'intention de l'interlocuteur et les émotions de base (Li *et al.*, 2017b).

Le corpus des nouvelles de *Maluuba* est un ensemble de données permettant de développer des agents conversationnels capables de répondre à des questions qui nécessitent des compétences de compréhension et de raisonnement au niveau humain. Cet ensemble de données construit à partir des articles de nouvelles de CNN possède 120K paires de requête/réponse. Les requêtes sont écrites par des humains en langage naturel (corpus dialogue long) (Trischler *et al.*, 2017).

2.8.2 Corpus dialogue axé sur les buts

Dans ce type de dialogues, les sujets des conversations n'appartiennent en général qu'à un seul domaine. Le corpus de dialogue Maluuba axé sur l'objectif est conçu pour aider à mener des recherches permettant de créer des agents de conversation capables

de prendre des décisions complexes. Cet ensemble de données a été préparé à l'aide de conversations interhumaines via une interface de discussion. Un humain a joué le rôle de client et l'autre a joué le rôle de l'agent de voyage (Asri *et al.*, 2017).

2.9 Évaluation des agents conversationnels

Les systèmes conversationnels sont évalués de manière semi-formelle, car le dialogue dépend fortement du contexte. La théorie présente du dialogue n'est pas assez précise pour spécifier une sortie cible à l'avance. L'évaluation actuelle utilise surtout les jugements humains pour estimer la cohérence de la conversation et corréler ces jugements avec des mesures extraites du système.

2.9.1 Évaluation automatique

On peut observer que les mesures automatiques ont presque toutes été inspirées par des mesures qui ont été utilisées avec succès dans les tâches de traduction automatique (BLEU (Papineni *et al.*, 2002), METEOR (Lavie et Agarwal, 2007)) et de résumés automatiques (ROUGE (Lin, 2004)). Les mesures de chevauchement des mots comme BLEU comparent la réponse d'un agent à une réponse humaine (Liu *et al.*, 2016). Cette métrique est la plus couramment utilisée dans la littérature sur la génération de réponses. Cependant, dans l'étude comparative de Liu *et al.* (2016), il a été prouvé que « *de nombreux paramètres couramment utilisés dans la littérature pour l'évaluation des systèmes de dialogue sans supervision ne sont pas en corrélation forte avec le jugement humain*». Il est largement débattu à quel point les mesures automatiques sont en corrélation avec une qualité de réponse, par exemple, ce que des nombres absolus signifient et quels sont les aspects de la qualité qu'ils mesurent ? (Liu *et al.*, 2016).

Plusieurs autres métriques d'évaluation ont été adaptées pour la tâche de génération de dialogue. Les plus utilisées sont basées sur les plongements de mots, à savoir :

⋆ La correspondance gourmande (*greedy matching*) : C'est la seule métrique basée sur le plongement des mots qui ne calcule pas son score au niveau de la phrase. Dans ce cas, la phrase n'est pas représentée par un vecteur. Au lieu de cela, étant donné deux phrases r et $\hat{r}$, on calcule la similitude cosinus de chaque vecteur (e_w) de mot $w \in r$ avec le plus proche mot $\hat{w} \in \hat{r}$. Ensuite, le score total est moyenné sur tous les mots :

$$G(r,\hat{r}) = \frac{\sum_{w \in r;} max_{\hat{w} \in \hat{r}} cos(sim(e_w, e_{\hat{w}}))}{\overline{r}} \tag{2.3}$$

Cette formule étant non symétrique ; nous devons faire la moyenne des scores d'appariement gourmands G dans les deux sens (Rus et Lintean, 2012a).

$$GM(r,\hat{r}) = \frac{G(r,\hat{r}) + G(\hat{r},r)}{2} \tag{2.4}$$

⋆ La métrique moyenne des plongements (*embedding average*) : il s'agit d'une méthode de calcul de la signification des phrases en faisant la moyenne des représentations vectorielles des mots composants (Mitchell et Lapata, 2008). La moyenne de plongements, e_w pour une phrase r, est définie comme suit (Liu *et al.*, 2016) :

$$\overline{e}_r = \frac{\sum_{w \in r} e_w}{|\sum_{w' \in r} e_{w'}|} \tag{2.5}$$

Pour comparer une réponse cible r avec une réponse source $\hat{r}$, nous calculons la similitude cosinus entre leurs plongements respectifs au niveau de la phrase (Liu *et al.*, 2016) :

$$EA = cos(\overline{e}_r, \overline{e}_{\hat{r}}). \tag{2.6}$$

⋆ Les vecteurs des extrêmes (*extrema vectors*) : c'est une façon de calculer les plongements au niveau des phrases. Elle consiste à utiliser les extrêmes vectoriels (Forgues *et al.*, 2014). Pour chaque dimension de vecteurs de mots, on sélectionne la valeur maximale parmi tous les vecteurs de mots d'une phrase de la façon suivante :

$$e_{rd} = \begin{cases} max_{w \in r}\ \mathrm{e}_{wd} \text{ si } e_{w'd} > |min_{w' \in r} e_{w'd}| \\ min_{w \in r}\ \mathrm{e}_{wd} \text{ sinon} \end{cases} \tag{2.7}$$

où d est l'index d'un vecteur ; e_{wd} est la dimension de e_w (plongement de mot w). Le min dans cette équation fait référence à la sélection de la plus grande valeur négative et ce si elle a une magnitude plus grande que la plus grande valeur positive dans une dimension d.

Ensuite, nous calculons la similitude entre le vecteur extrême source et cible de réponse en utilisant la distance cosinus. Par intuition, cette approche donne la priorité aux mots informatifs par rapport aux mots courants ; les mots qui apparaissent dans des contextes similaires seront rapprochés dans l'espace vectoriel. Ainsi, les mots communs sont tirés vers l'origine de l'espace vectoriel, car ils se produisent dans divers contextes, tandis que les mots porteurs d'informations sémantiques importantes vont se retrouver plus loin. En prenant les extrêmes le long de chaque dimension, nous sommes donc plus susceptibles d'ignorer les mots courants (comme les mots vides ou *stop word*, en anglais) (Liu *et al.*, 2016).

D'autres métriques souvent utilisées dans les tâches séquence vers séquence, ont été étendues vers les tâches de génération de dialogue. Un exemple est la perplexité des mots, une métrique automatique utilisée pour évaluer les modèles de dialogue (Pietquin et Hastie, 2013a; Serban *et al.*, 2016b). Celle-ci mesure la façon dont le modèle prédit des réponses obtenues au moment du test. Étant donné un contexte c et une réponse $r = w_1, \cdots, w_n$, la perplexité est définie comme suit :

$$Perplexité(c,r) = \frac{1}{\sqrt[n]{p(w_1, w_2, \cdots, w_n|c)}} = exp\left\{-\frac{1}{n}\sum_{i=1}^{n} log\ p(w_i|w_1, \cdots, w_{i-1}, c)\right\} \tag{2.8}$$

La perplexité est considérée comme une mesure utile car plus le score est bas, plus le modèle prédit des réponses obtenues avec précision et capture la distribution de l'ensemble de données.

2.9.2 Évaluation manuelle

D'après Liu *et al.* (2016), les évaluations humaines devraient être utilisées pour l'évaluation de la qualité des agents conversationnels, bien que coûteuses et non reproductibles. Ce type d'évaluations peut-être accompagné d'évaluations basées sur des mesures automatiques. Des initiatives telles que la plate-forme MTurk [14] aident à accomplir des évaluations humaines de façon collaborative, sous forme de jugements de pertinence (Pietquin et Hastie, 2013b).

Dans la littérature, la méthode la plus courante pour évaluer les agents conversationnels est la comparaison entre des paires de réponses à travers des jugements humains. Dans ces évaluations, un contexte et deux réponses sont présentés devant des juges humains qui sont invités à choisir les réponses pertinentes, (Vinyals et Le, 2015; Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Li *et al.*, 2016c; Sordoni *et al.*, 2015b; Park *et al.*, 2018; Xing *et al.*, 2018). On note que les instructions sont relativement vagues. On demande généralement aux juges quelle réponse ils préfèrent ou quelle réponse est la meilleure. Cette méthode est le plus utilisée dans la littérature, car l'écart entre les évaluations des juges est très faible. Une autre méthode a été créée en utilisant des scores absolus afin d'évaluer la qualité des modèles (Serban *et al.*, 2017a). Dans cette méthode, les évaluateurs humains attribuent un score absolu sur une échelle comprise entre 0 et 4, où 0 correspond à la réponse de mauvaise qualité et 4 à la réponse de bonne qualité.

Un autre paradigme est l'évaluation par réseaux antagonistes génératifs (Generative Adversarial Networks (GANs)) (Bowman *et al.*, 2016; Li *et al.*, 2017a; Bruni et Fernandez, 2017), inspirée du test de Turing. L'idée est de former un classifieur d'évaluation pour faire la distinction entre les réponses générées par l'homme et les réponses générées par un modele de dialogue (Générateur). Ce modele de distinction est appelé

14. Amazon Mechanical Turk Crowdsourced Marketplace

discriminateur. Les solutions proposées ont des problèmes majeurs comme la dépendance au corpus d'entraînement. En outre, il n'y a aucune garantie que ce classifieur est un bon évaluateur.

2.10 Traduction Automatique Neuronale et Optimisation du Transformer

La Traduction Automatique Neuronale (TAN) comme son nom l'indique est une technologie basée sur les réseaux de neurones artificiels pour prédire la probabilité d'une séquence de mots, modélisant ainsi des phrases dans un seul modèle intégré (Kalchbrenner et Blunsom, 2013). La TAN a fait des progrès considérables, surtout grâce à l'apprentissage profond et plus spécifiquement aux réseaux de neurones récurrents (Kalchbrenner et Blunsom, 2013). Sutskever *et al.* (2014) ont créé l'architecture séquence vers séquence avec l'utilisation du LSTM. Ensuite, ce modèle a été augmenté par l'utilisation de l'attention comme mécanisme sélectif (Bahdanau *et al.*, 2015). Initialement, ce mécanisme a été présenté sous une forme générale puis a été simplifié par Luong *et al.* (2015b).

D'autres types d'architectures sont apparus en utilisant des blocs convolutifs et le mécanisme d'attention, comme ConvSeq2Seq (Gehring *et al.*, 2017). Afin de maintenir l'ordre de la séquence, les auteurs ont ajouté un mécanisme d'ordonnancement explicite nommée «encodage positionnel». Une autre amélioration vient de s'ajouter à cette architecture avec le modèle DynamicConv (Wu *et al.*, 2019), qui utilise plusieurs noyaux de convolution en parallèle au lieu d'utiliser un seul noyau par couche dans ConvSeq2Seq.

Toujours récemment, Vaswani *et al.* (2017) ont introduit le Transformer, une architecture basée entièrement sur le mécanisme d'attention (Secic *et al.*, 2019; Zhang *et al.*, 2020a). Le Transformer a été proposé comme alternative avantageuse à l'approche basée sur RNR afin d'implémenter des architectures encodeur-décodeur pour la traduction

de séquences, telles que trouvées dans les systèmes de traduction (Tong et Yen, 2014; Dušek, 2017), suggestion de requête (Sordoni *et al.*, 2015b), génération de dialogue (Serban *et al.*, 2016b; Zhang *et al.*, 2019a), etc. Les résultats de la tâche partagée dans l'édition 2019 de la conférence sur la traduction automatique (WMT2019) révèlent une croissance constante de l'utilisation des architectures basées sur Transformer (Barrault *et al.*, 2019b).

Initialement, le modèle Transformer fut introduit pour le traitement automatique du langage naturel (Wolf *et al.*, 2020). Par la suite, il a été étendu à de nombreux autres domaines tels que la génération musicale (Zhang, 2020) et le traitement d'images (Parmar *et al.*, 2018, 2019).

Afin de profiter de ses avantages, le Transformer a été utilisé dans la formation de modèles linguistiques extrêmement volumineux qui vise à transférer des connaissances vers d'autres tâches cibles (Devlin *et al.*, 2019; Radford *et al.*, 2019b; Xia *et al.*, 2020). En 2018, la compagnie OpenAI a publié le modèle de langage GPT (Generative Pre-trained Transformer) qu'elle a entraîné sur des données textuelles de Wikipédia (Radford *et al.*, 2018). Ensuite, une version améliorée, GPT-2, a été proposée avec un entraînement sur des données de Reddit, dix fois plus grandes que les précédentes (Radford *et al.*, 2019b). La version GPT-3 a suivi en augmentant le corpus d'entraînement avec les données WebText et en utilisant des représentations clairsemées (sparse representations) des neurones pour accélérer l'apprentissage. Il utilise 12 couches d'encodage et est dédié à l'apprentissage par transfert (Brown *et al.*, 2020). L'accès à GPT-3 est fourni exclusivement via une API de Microsoft (Brown *et al.*, 2020).

Dans la même période, Devlin *et al.* (2019) de Google AI ont créé le modèle BERT [15] pour la compréhension des langages naturels afin d'améliorer leur système de recherche pour diverses langues. La différence entre ce modèle et celui de Transformer est l'ajout

15. BERT : Bidirectional Encoder Representations from Transformers

d'un plongement de segment au plongement des mots et les vecteurs de position. Un autre point important de BERT est son mécanisme d'apprentissage avec le remplacement de certains mots par le mot *MASK*, ce qui lui permet d'être bidirectionnel. Contrairement à la plupart des modèles unidirectionnels comme GPT, où on leur donne le début d'une phrase et ils doivent prédire le prochain mot en apprenant uniquement du contexte gauche de la phrase. Dans BERT, on masque un mot au milieu de la phrase et on doit essayer de le prédire en ayant accès à la fois au contexte gauche et au contexte droit. Cette technique est nommée modèle de langage masqué (MLM) (Xia *et al.*, 2020). Ce modèle a bénéficié de plusieurs améliorations, dont ALBERT (A Lite BERT) qui utilise une taille de plongement relativement faible par rapport de celle de BERT (Lan *et al.*, 2020). Une autre version proposée est RoBERTa (A Robustly Optimized BERT Approach) qui utilise un masque dynamique durant l'entraînement (Joshi *et al.*, 2020).

Liu *et al.* (2020) ont proposé une nouvelle couche d'encodage de position différente de l'encodage sinusoïdal statique. Cette couche contient des paramètres pouvant être appris durant l'entraînement afin de le rendre dynamique. Cette couche de position peut s'adapter à différents jeux de données et à différentes architectures.

Fan *et al.* (2020) ont proposé une variante simple du transformer appelée transformer attentif multibranches (brièvement, MAT), inspirée des tâches de vision par ordinateur (Zhang *et al.*, 2019b), où la couche d'attention est la moyenne de plusieurs branches et chaque branche est une couche d'attention multitêtes indépendante. Cette méthode est similaire au RNR-bidirectionnel que l'on peut considérer comme un RNR a deux branches. (Fan *et al.*, 2020)).

Carreto Fidalgo *et al.* (2021) ont proposé une méthode appelée swap-then-finetune qui utilise l'apprentissage par transfert et remplace l'attention softmax utilisée par le décodeur par une alternative RNR. Cette approche améliore le temps d'entraînement et la précision par rapport au transformer de base. Cependant, elle demande un modèle

Transformer pré-entraîné.

Tay *et al.* (2021) ont proposé l'architecture OmniNet, où des représentations omnidirectionnelles sont utilisées par le Transformer. Ainsi, au lieu de maintenir un champ réceptif strictement horizontal pour calculer l'attention, chaque jeton est comparé avec tous les jetons de l'ensemble du réseau et dans toutes les profondeurs de réseau. Ce processus d'attention omnidirectionnelle est très coûteux en complexité de calcul à cause du nombre de couches ajoutées.

Un autre modèle de Transformer a été présenté sous une forme hiérarchique afin de traiter plusieurs langues proches. Ces langues partagent des vocabulaires et des règles linguistiques qui sont utilisés afin de créer un arbre linguistique qui montre leurs degrés de parenté. En général. Cette nouvelle approche du partage de paramètres dans la traduction automatique multilingue a été suggérée récemment par (Khusainova *et al.*, 2021).

Compte tenu des énormes demandes de calcul des modèles de traitement de séquence, il y a un besoin croissant de trouver des méthodes d'optimisation pouvant rendre le modèle plus profond et réduire les besoins en mémoire et calcul du Transformer. Ces besoins dépassent ce que les méthodes standards peuvent offrir, comme la réduction de précision et les points de contrôle du gradient (Shazeer et Stern, 2018; Sohoni *et al.*, 2019). Deux types d'optimisation de modèle se retrouvent dans la littérature : Celui qui tente de simplifier la représentation des données en mémoire (par exemple, (Dai *et al.*, 2019; Correia *et al.*, 2019)) et celui qui tente de réduire la complexité de calcul du modèle (par exemple, (Guo *et al.*, 2019; Kitaev *et al.*, 2020)).

Récemment, des versions plus efficaces du mécanisme d'auto-attention du Transformer ont été explorées (Sukhbaatar *et al.*, 2019). Parmi elles, on trouve celles qui s'avèrent prometteuses en exploitant l'éparcité dans les couches d'attention, conduisant au Sparse Transformer (Correia *et al.*, 2019). Cette optimisation exploite une représentation clair-

semée, factorisée de la matrice d'attention afin de réduire l'utilisation de la mémoire et du calcul en $O(N\sqrt{N})$ par rapport à la complexité de transformer de base qui est $O(N^2)$. Cette technique est utilisée dans GPT-3 (Radford *et al.*, 2019b).

Guo *et al.* (2019) présentent le Star-Transformer comme une alternative légère, obtenue par éparcification du calcul des noeuds neuronaux. Pour réduire la complexité du modèle, les auteurs remplacent la structure entièrement connectée des attentions par une topologie en étoile, où le calcul de l'attention est limité aux voisins des mots. Cette simplification est adaptée pour la tâche de détection des entités nommées dont le contexte entre les mots est très proche.

Dai *et al.* (2019) ont proposé le Transformer-XL qui utilise un mécanisme de récurrence au niveau des segments pour l'apprentissage des séquences variables. À cette fin, le modèle introduit un nouveau schéma d'encodage de la position relative qui rend le modèle dynamique par longueur de phrase (Huang *et al.*, 2021).

Enfin, Kitaev *et al.* (2020) ont proposé le modèle Reformer, qui réduit l'effort de calcul du Transformer en utilisant l'algorithme de hachage sensible à la localité (*Locality Sensitive Hashing-LSH*) pour optimiser le calcul de l'attention. Cet algorithme utilise la recherche du plus proche voisin en se basant sur la distance cosinus entre les vecteurs clés et requêtes. Cela a permit d'optimiser le calcul de l'attention en réduisant sa complexité de $O(N^2)$ à $O(Nlog(N))$, N étant la longueur de la séquence. Les auteurs ont aussi proposé l'intégration des couches résiduelles réversibles pour une amélioration supplémentaire du temps de traitement et de la mémoire, de la même façon que le modèle de RevNet (Gomez *et al.*, 2017).

L'attention du Transformer a des problèmes pour calculer les longues séquences, en raison que le calcul d'auto-attention évolue quadratiquement avec la longueur d'une séquence. Pour y remédier, Beltagy *et al.* (2021) introduisent le Longformer. Ce modèle utilise un mécanisme d'attention qui évolue linéairement avec la longueur de la

séquence, ce qui facilite les traductions des longs documents avec des milliers de jetons ou plus. Le mécanisme du Longformer remplace l'auto-attention standard par plusieurs attentions locales appliquées sur des fenêtres de taille fixe. Cette méthode a été inspirée de la convolution utilisée dans les réseaux de neurones convolutifs (Simard *et al.*, 2003). L'optimisation proposée pour ce modèle vise principalement les documents longs ; pour les textes courts, la performance est très proches de celle du transformer, car la taille de ces fenêtres devient proche de celle des textes (Beltagy *et al.*, 2021).

Dans l'état-de-l'art présenté, nous notons que la réduction du nombre de paramètres du mécanisme d'attention du Transformer n'est pas abordée. De ce fait, notre proposition de réduire le nombre de paramètres de l'attention est novatrice. Par ailleurs, nous appliquons notre approche à la génération de dialogues multi-tours.

2.11 Évaluation de la traduction automatique

Dans cette section, nous présentons les métriques d'évaluation automatique pour la traduction qui repose généralement sur une comparaison entre les mots de l'hypothèse et la référence.

2.11.1 Taux d'erreurs de mots

L'une des métriques d'évaluation la plus populaire pour la traduction automatique est le taux d'erreurs de mots (en anglais : *Word Error Rate* ou *WER*).

Cette métrique est dérivé de la distance de Levenshtein, en utilisant les mots au lieu des caractères. Elle indique le taux de mots incorrectement reconnus par rapport à un texte de référence, et définie comme suit (Popović et Ney, 2007) :

$$\text{WER} = \frac{\#Ins + \#Sub + \#Del}{\#\text{Total de mots dans la référence}} \times 100\% \tag{2.9}$$

où *#Ins* désigne le nombre de *mots* alignés, qui sont ajoutés dans l'hypothèse, *#Sub* est

le nombre de mots de référence, qui sont remplacés par les mots alignés dans l'hypothèse, *#Del* est le nombre de mots, qui n'existent pas dans la référence (Popović et Ney, 2007).

Cette métrique compte généralement le nombre d'opérations d'édition nécessaire afin de recréer la référence à partir de l'hypothèse.

2.11.2 La métrique BLEU

La métrique BLEU [16] sert à évaluer la qualité d'une réponse par un système de traduction (Papineni *et al.*, 2002). Cette métrique définit $P_1, P_2, P_3, \cdots P_N$ comme précisions de n-gramme (proportion de n mots successifs prédits correctement par rapport aux N mots d'une référence).

Ensuite, elle calcule la moyenne géométrique de toutes les précisions prises ensemble, appelée G_{mean}, des ratios donnés comme suit :

$$G_{mean} = \left(\prod_{i=1}^{n} P_i \right)^{\frac{1}{n}} = exp\left(\frac{1}{n} \sum_{i=1}^{n} log\ p_i \right) \tag{2.10}$$

Une pénalité de brièveté (brevity penalty ou BP) et alors introduite, qui calcule la corrélation entre la longueur de référence r et la longueur d'hypothèse c :

$$BP = \begin{cases} 1 & \text{si } c > r \\ exp(1 - \frac{r}{c}) & \text{sinon} \end{cases} \tag{2.11}$$

Finalement, la métrique BLEU est calculée comme suit :

$$BLEU = BP\ X\ G_{mean} \tag{2.12}$$

16. BLEU : Bilingual Evaluation Understudy

2.11.3 Métrique d'évaluation de traduction avec l'ordonnancement explicite (METEOR)

La métrique d'évaluation de traduction avec l'ordonnancement explicite *METEOR* [17], proposée par Lavie et Agarwal (2007), consiste à mieux corréler les hypothèses de traduction avec les jugements humains, en utilisant davantage des alignements des mots entre une hypothèse et quelques références. L'alignement est basé sur trois modules : le premier utilise une correspondance exacte entre les formes des mots, le second compare les racines des mots (en anglais : *stems*) et le troisième utilise les synonymes venant de ressources lexicales externes, telle que *WordNet* (Miller, 1995) ou *BabelNet* (Navigli et Ponzetto, 2012).

Formellement, le score de *METEOR* utilise la moyenne harmonique $F_{measure}$ (Calculer par la précision et le rappel) et un facteur additionnel, appelé *Pénalité* (en anglais : *Penalty*), comme suit :

$$F\text{-}measure = \frac{P \times R}{\alpha \times P + (1-\alpha) \times R} \; 0 \leq \alpha \leq 1 \tag{2.13}$$

$$Penalty = \gamma \times \left(\frac{\text{number of chunks}}{\text{number of matches}} \right)^{\beta} \; 0 \leq \gamma \leq 1, \beta \in \mathbb{N} \tag{2.14}$$

$$METEOR\ score = (1 - Penalty) \times F\text{-}measure \tag{2.15}$$

La précision $P = m/c$ et le rappel $R = m/r$, où c et r sont respectivement des longueurs des candidates et des références. m est le nombre d'unigrammes trouvé entre les deux textes.

number of chunks est le plus petit nombre possible de segments de mots partagés entre la phrase de l'hypothèse et de la référence, *number of matches* est le nombre d'unigrammes qui composent la référence.

Les paramètres $0 \leq \alpha \leq 1, 0 \leq \gamma \leq 1$, ,$\beta \in \mathbb{N}$, sont des facteurs réglés pour maximiser la

17. METEOR : *Metric for Evaluation of Translation with Explicit ORdering*

corrélation entre cette métrique et les jugements humains (Denkowski et Lavie, 2011).

2.12 Conclusion

Dans ce chapitre, nous avons présenté une vision générale des travaux concernant les différentes approches relatives à la conception d'un agent de conversation, la classification et la structure de ces agents conversationnels. Nous avons aussi couvert l'évolution, au fil des ans, des techniques de conception ayant été adoptées pour mettre en oeuvre différents types d'agents de conversation. Nous avons également présenté un état de l'art des agents génératifs multi-tours en mettant en évidence les faiblesses des systèmes actuels, ainsi que les types des corpus de dialogues utilisés.

Nous concluons que les agents de conversation ont maintenant parcouru un long chemin depuis les approches basées sur les règles, la recherche par motifs, la récupération simple jusqu'à l'apprentissage profond.

Dans ce chapitre, nous avons aussi présenté un aperçu d'un état-de-l'art de la traduction automatique, comme deuxième tâche du TAL étudiée dans cette thèse, avec les différentes métriques d'évaluations utilisées.

Dans le chapitre qui suit, nous présentons la méthodologie proposée pour traiter les problèmes abordés dans cette thèse. Le premier modèle proposé est appliqué à la tâche de traduction automatique, et tente de réduire le nombre de paramètres utilisés dans le Transformer de base. Le deuxième modèle proposé utilise une attention multidimensionnelle avec une application aux systèmes de génération de dialogue (Belainine *et al.*, 2022a). Le troisième modèle proposé fusionne les deux afin de tirer parti de leurs fonctionnalités respectives (Belainine *et al.*, 2022b).

CHAPITRE III

MÉTHODOLOGIE

3.1 Introduction

Ce chapitre présente la démarche méthodologique que nous avons adoptée pour la création d'agents conversationnels et le cadre expérimental que nous avons mis en pratique pour valider les solutions proposées. La figure 3.1 donne un aperçu de notre méthode. La méthodologie suivie comprend deux volets qui sont combinés à la fin pour atteindre notre objectif final. Dans un premier temps, nous présentons un modèle de traduction automatique qui n'inclut pas de RNR dans sa composition, en partant de l'architecture Transformer. Notre contribution est le remplacement de l'attention requête-clé-valeur du Transformer par une attention basée sur la similarité cosinus. Cette architecture nous a donné des résultats meilleurs que ceux du modèle original.

La deuxième phase reprend l'architecture encodeur-décodeur à base de RNR pour la génération de dialogues, mais l'améliore en utilisant un traitement en cascade (Belainine *et al.*, 2022a), où intervient un encodeur et plusieurs décodeurs, le tout lié par un mécanisme d'attention multidimensionnelle. Ce modèle élimine l'encodage hiérarchique utilisé précédemment (dans l'état-de-l'art), réduisant ainsi les risque d'amplification et de propagations d'erreurs lors de l'apprentissage.

La troisième phase reprend l'attention multidimensionnelle pour l'appliquer au pre-

mier modèle et ainsi étendre sa fonctionnalité à la génération de dialogues multi-tours (Belainine *et al.*, 2022b). Notre contribution inclut un nouvel encodage de position qui permet une architecture à un seul encodeur et un seul décodeur, avec un performance supérieure à plusieurs modèles de l'état de l'art. De plus, le modèle obtenu peut service autant à la traduction automatique qu'à la génération de dialogues.

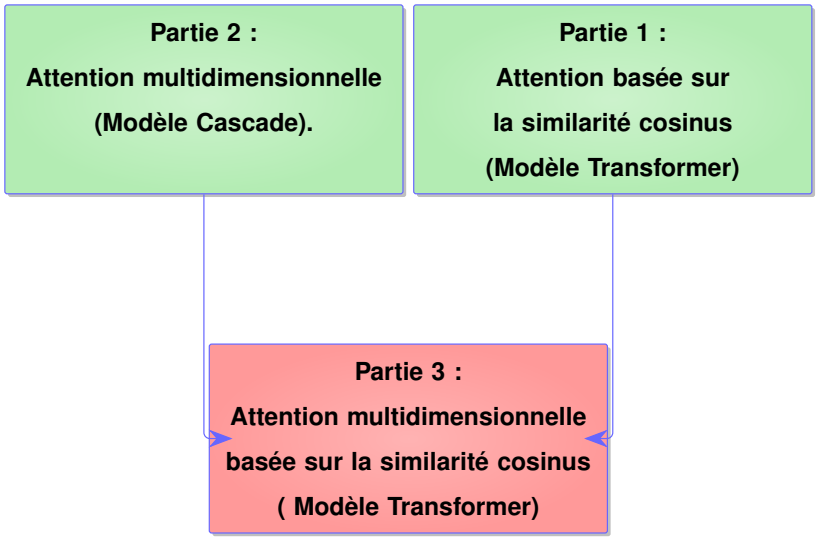


Figure 3.1: Combinaison architecturale de la démarche méthodologique.

3.2 Transformer à base de similitude cosinus

Dans cette section, nous décrivons notre proposition d'un mécanisme d'attention basé sur la similarité cosinus et qui utilise deux matrices de représentation au lieu de trois dans le Transformer. Concrètement, la représentation clé-valeur du Transformer est remplacée par une seule représentation et le calcul d'attention du Transformer, qui utilise la similarité produit scalaire, utilise la similarité cosinus dans le modèle modifié. Ainsi, nouvelle attention repose sur la projection des plongements de mots dans les séquences d'entrée dans deux espaces de représentation $\mathbb{Q}$ et $\acute{\mathbb{Q}}$, le premier étant relatif à la séquence d'entrée et le second à la séquence cible. La similarité cosinus est alors utilisée pour déterminer la façon dont les deux représentations sont liées l'une à

l'autre et le résultat est utilisé pour calculer différents types d'attention comme le fait le Transformer. Comme montré plus bas (sous-section 3.2.5), ces modifications à l'architecture du Transformer réduisent sa consommation en temps et en mémoire ainsi que le nombre de paramètres utilisés. De plus, le nouveau modèle n'utilise pas le facteur d'échelle empirique ou les approximations que l'on retrouve dans le Transformer et le Reformer.

3.2.1 Modèle théorique

D'une façon générale, le problème de transduction de séquence est modelé comme une distribution de probabilité conditionnelle P avec des paramètres θ tels que :

$$P_\theta(Y|X) = P_\theta(y_1|X) \prod_{m=2}^{M} P_\theta(y_m|y_{<m}, X) \tag{3.1}$$

Où X est une séquence d'entrée, Y une séquence de sortie, et $y_{<m} = y_1, \cdots, y_{m-1}$ sont les éléments de la séquence de sortie avant l'élément y_m.

Comme le montre l'équation 3.1, chaque élément de Y est généré en considérant la séquence d'entrée et l'historique de sortie comme facteurs probabilistes. À cet égard, un contexte dynamique c_m peut être créé par la combinaison de X et $y_{<m}$, de sorte que nous avons :

$$P_\theta(y_m|y_{<m}, X) = P_\theta(y_m|c_{m-1}) \tag{3.2}$$

Ensuite, un modèle génératif simple, inspiré de Mikolov *et al.* (2010), peut être construit en appliquant la fonction $Softmax$ à une mesure de la relation entre c_{m-1} et les valeurs potentielles de y_m (Serban *et al.*, 2017c) :

$$P_\theta(y_m = v|c_{m-1}) = \frac{exp\ o_v^T c_{m-1}}{\sum_k exp\ o_k^T c_{m-1}} \tag{3.3}$$

Dans cette équation, o_v est un vecteur à valeurs réelles qui représente le plongement

de mot d'une valeur potentielle de y_m, avec sa probabilité à la position m atteignant un maximum lorsqu'il est colinéaire avec c_{m-1}. Les paramètres du modèle sont appris par l'algorithme de descente de gradient, en utilisant la perte d'entropie croisée (Moher, 1993).

Les équations précédentes sont typiques des modèles de transduction de séquence de type encodeur-décodeur, dont le Transformer (Vaswani *et al.*, 2017) où une séquence de plongements des mots $w = w_1, ..., w_M$ à l'entrée sert à générer une séquence de représentations basées sur l'attention $z = z_1, ..., z_M$ afin d'aider le décodeur à générer la séquence de sortie. Comme le montre le côté gauche de la figure 3.2, ceci est réalisé en empilant plusieurs couches de traitement dont chacune comprend essentiellement une ou de deux sous-couches d'attention suivies d'un réseau neuronal à propagation directe (feedforward) peu profond. Cette structure à N couches utilisée à la fois dans l'encodeur et dans le décodeur. De plus, plusieurs perspectives sont utilisées pour calculer l'attention dans chaque couche, conduisant ainsi à un mécanisme d'attention à H têtes. L'article original du Transformer mentionne $N = 6$ et $H = 8$ comme valeurs de base.

Nous utilisons un flux de traitement similaire dans ce travail, avec des optimisations apportées pour une meilleure efficacité et précision de calcul, et une réduction du nombre de paramètres comme déjà mentionné.

3.2.2 Pile d'encodeur et de décodeur

L'architecture du Transformer comprend deux modules qui sont l'encodeur et le décodeur, avec pour but de résoudre les problèmes de type séquence à séquence. Chaque sous-module est constitué d'un empilement de couches unitaires. Une couche unitaire d'encodeur ou de décodeur est construite sur la base d'attentions locales (auto-attentions). Son rôle est de déterminer les relations entre les mots d'une phrase. La seule différence entre les couches unitaires de l'encodeur et du décodeur est que la der-

nière inclut un autre type d'attention nommé attention croisée. Le rôle de cette attention est de déterminer la relation entre les mots des phrases source et cible.

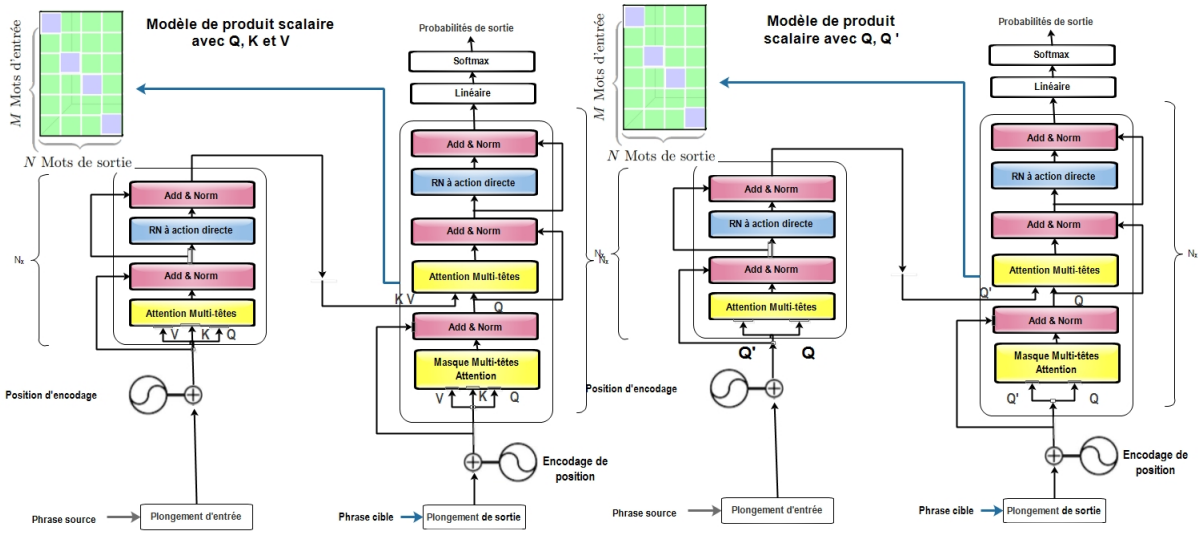


Figure 3.2: Illustration de l'architecture du transformer (Vaswani *et al.*, 2017) (à gauche) et de notre modèle (à droite)

Dans le Transformer original, les attentions utilisent des couches basées sur le produit scalaire mis à l'échelle qui a été formulée par les matrices Q, K et V représentant une attention de type requête-clé-valeur. La formule génerale d'attention est définie par (Vaswani *et al.*, 2017) :

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.4}$$

Dans notre modèle, nous avons remplacé les matrices Q, K et V par deux matrices Q et $\acute{Q}$ comme illustré à la figure 3.2, dans le but de remplacer l'approche de produit scalaire mis à l'échelle pour le calcul d'attention par une approche basée sur la similarité cosinus. Ces changements sont décrits dans la prochaine sous-section.

3.2.3 Représentation des données avec l'attention

Notre modèle projette chaque plongement de mot w_i à l'entrée dans deux vecteurs de représentation q_i et $\acute{q}_i$, l'un représentant l'espace source et l'autre l'espace cible. Ces vecteurs alors normalisés par leurs normes géométriques $\|q_i\|$ et $\|\acute{q}_i\|$.

Afin de calculer un score de similarité cosinus entre deux vecteurs qui représentent les similarités entre les mots dans les mêmes phrases. Ces vecteurs doivent être normalisés afin de rendre le produit scalaire indépendant de la longueur de ces vecteurs. Notre modèle projette chaque plongement de mot w_i à l'entrée dans deux vecteurs de représentation q_i et $\acute{q}_i$, l'un représentant l'espace source et l'autre l'espace cible. Ces vecteurs alors normalisés par leurs normes géométriques $\|q_i\|$ et $\|\acute{q}_i\|$. Ensuite, la sumilarité cosinus entre deux mots w_i et w_j est obtenu par le produit entre $q_i/\|q_i\|$ et $\acute{q}_j/\|\acute{q}_j\|$.

Afin d'effectuer cette opération dans un lot de données, deux matrices Q et $\acute{Q}$ sont obtenues en empilant les vecteurs de représentation q_i et $\acute{q}_i$ obtenus d'une même phrase, et la similitude cosinus des deux matrices est calculée par le produit des deux matrices avant d'appliquer la fonction $Softmax$ au résultat. Finalement, on multiplie le résultat trouvé par la matrice $\acute{Q}$ afin de pondérer ses éléments :

$$\begin{aligned} Attention(Q,\acute{Q}) &= Softmax\left(cos\left(Q\acute{Q}^T\right)\right)\frac{\acute{Q}}{\|\acute{Q}\|} \\ &= Softmax\left(\frac{Q\acute{Q}^T}{\|Q\|\|\acute{Q}^T\|}\right)\frac{\acute{Q}}{\|\acute{Q}\|} \end{aligned} \tag{3.5}$$

Tel que $\|\acute{Q}\| = [\|\acute{q}_0\|,\cdots,\|\acute{q}_n\|]$ et $\|Q\| = [\|q_0\|,\cdots,\|q_n\|]$ sont les vecteurs des normes géométriques pour les vecteurs des matrices $\acute{Q} = [\acute{q}_0,\cdots,\acute{q}_n]$ et $Q = [q_0,\cdots,q_n]$, et n est la taille de la séquence d'entrée.

Puisque l'argument de la fonction $Softmax$ est borné par l'image du cosinus, il prend des valeurs dans l'intervalle [-1, 1] et il n'est pas nécessaire de le mettre à l'échelle comme fait le Transformer pour le produit scalaire de l'équation 3.4.

Nous appelons notre approche « appariement d'espaces direct » (*Direct Space Matching ou DSM*) pour refléter la suppression de la matrice de clés intermédiaire utilisée dans le Transformer et le Reformer. Le produit scalaire de Q et $\acute{Q}$ dans l'équation 3.21 aide à déterminer directement les symboles cibles correspondant le mieux aux symboles source.

La figure 3.12 illustre notre attention DSM par rapport au produit scalaire mis à l'échelle utilisé dans le Transformer et le Reformer. Elle montre le mappage linéaire par deux matrices apprises W_q, $W_{\acute{q}}$ des vecteurs de plongements à l'entrée. Le résultat est les représentations q, $\acute{q}$ suivi par la normalisation géométrique des deux vecteurs. Le produit scalaire entre les vecteurs normalisés, qui en résulte, est représenté par une similarité cosinus. Ensuite, on multiplie le résultat par $\acute{Q}$ afin de la pondérer.

3.2.4 Attention multi-têtes

Nous avons adapté le modèle de Transformer aux représentations q et $\acute{q}$. Par conséquent, chaque plongement d'entrée de taille $d_{modèle}$ est mappé à h paires de vecteurs q et $\acute{q}$ de dimensions $d_q = d_{\acute{q}} = d_{modèle}/h$, et l'attention est calculée pour chaque paire en utilisant l'équation 3.21. Ensuite, les h résultats trouvés sont concaténés afin de produire un vecteur z de dimension $h \times d_q$. Cette concaténation est faite par un réseau de neurones à propagation direct afin de produire un résultat d'attention final de même dimension qu'a l'entrée, permettant ainsi d'utiliser la sortie de la couche actuelle comme entrée de la prochaine couche de l'encodeur ou du décodeur comme le montre la figure 3.2. Formellement, nous avons :

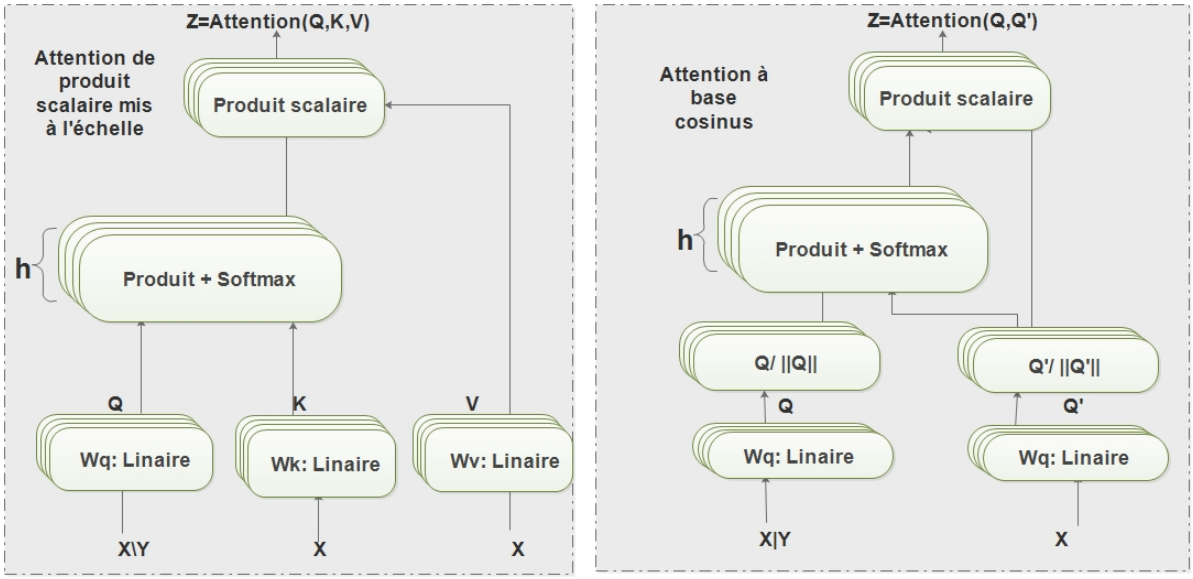


Figure 3.3: Illustration de notre modèle d'attention (à droite) par rapport à l'approche du produit scalaire mis à l'échelle utilisée dans le Transformer et le Reformer (à gauche)

$$\begin{cases} \text{Multi-têtes}(Q,\acute{Q}) = concat(\text{tête}_1, \text{tête}_2 \cdots \text{tête}_h)W^o \\ \qquad\qquad Avec \\ \text{tête}_i = attention(Q_i, \acute{Q}_i), i = 1,...,h \end{cases} \tag{3.6}$$

Les h paires de matrices qui relient chaque plongement d'entrée à ses représentations q et $\acute{q}$ sont $q = xW_i^Q$ et $\acute{q} = xW_i^{\acute{Q}}$ telles que $W_i^Q \in R^{d_{\text{modèle}} \times d_q}, W_i^{\acute{Q}} \in R^{d_{\text{modèle}} \times d_{\acute{q}}}$, et la matrice de projection dans l'équation 3.6 est telle que $W^O \in R^{h d_{\text{tête}} \times d_{\text{modèle}}}$.

Les têtes sont toutes calculées en parallèle, et le réglage $d_{\text{tête}} = d_{\text{modèle}}/h$ qui garde la dimension de l'attention multi-têtes similaire à celle de l'attention d'une tête unique.

Nous utilisons les mêmes paramètres que ceux utilisés par le modèle de base du Transformer, nombre de tête $h = 8$, $d_q = d_v = 64$ et la taille de la couche cachée de réseaux à action directe $d_{ff} = 2048$. Le processus d'apprentissage est itéré pour tous les mots sources et cibles dans l'ordre. La comparaison est l'objectif principal de l'utilisation du

même nombre de têtes que le Transformer.

3.2.5 Complexité du modèle

Le tableau 3.1 fournit la consommation de mémoire et le nombre d'opérations pour calculer l'attention dans chaque modèle, en supposant une taille de vecteur égale à ($d_q = d_k$) et une longueur de séquence égale à l. Comme le tableau le montre, le remplacement des vecteurs k et v dans le Transformer par un seul vecteur q' économise 1/3 de l'espace mémoire et 1/3 du temps de calcul du gradient durant l'entraînement, grâce à une réduction d'une multiplication vectorielle pour chaque passage en avant et d'une mise à jour en moins de la matrice de poids par la suite. EN revanche, l'équation 3.4 utilise une division par une constante contre deux divisions par une norme dans l'équation 3.21. Toutefois, cela n'a pas d'impact significatif sur la complexité de calcul $O(l^2)$ des deux équations.

Dans le modèle Reformer, on utilise la même architecture que celle du Transformer. Cependant, l'approximation LSH [1] et l'optimisation du calcul à travers les couches résiduelles réversibles lui permettent de réduire considérablement l'empreinte mémoire et la charge de calcul globale, mais cela est fait aux dépens d'une dégradation de la précision de prédiction (Kitaev *et al.*, 2020).

Le tableau 4.13 fournit la complexité totale de notre modèle, du Transformer et du Reformer, en utilisant la même notation adoptée par Kitaev *et al.* (2020). Ainsi, l est la longueur de la séquence d'entrée, $d_{modèle}$ la taille de vecteur de plongement, d_{ff} la

1. Locality sensitive hashing ou hachage sensible à la localité est un algorithme de résolution des problèmes liés à *la malédiction de la dimensionnalité* de l'espace. Son fonctionnement repose sur la recherche approximative en utilisant la méthode des plus proches voisins. L'idée clé est d'appliquer une fonction de hachage telle que les points proches ont une forte probabilité d'avoir la même valeur de hachage dans l'espace cible (Chakrabarti *et al.*, 2015)

	Equation	Mémoire	Temps d'attention	Temps Gradient
Transformer	$Softmax(\frac{QK^T}{\sqrt{d_k}})V$	$3d_ql$	d_ql^2	$3d_ql$
Notre modèle	$Softmax(cos(Q\acute{Q}^T))\frac{\acute{Q}}{\|\acute{Q}\|}$	$2d_ql$	d_ql^2	$2d_ql$

Tableau 3.1: Comparaison entre notre modèle et le Transformer en termes de consommation de la mémoire, du nombre d'opérations temporelles de l'attention et des calculs de gradient.

taille de la couche cachée du réseau de neurones à action directe, b la taille du lot, n_l le nombre de couches dans l'encodeur ou le décodeur, n_h le nombre de têtes d'attention. Pour le Reformer, n_c est le nombre des segments LSH et n_r le nombre de répétitions de hachage. Dans (Kitaev *et al.*, 2020), les auteurs utilisent $n_c = l/32$ de manière à avoir $4l/n_c = 128$. De plus, ils utilisent $c = 128$ pour le nombre de blocs dans les calculs anticipés (Kitaev *et al.*, 2020). Le tableau 4.13 confirme à nouveau l'avantage de notre modèle en termes de gain du temps et de réduction de la consommation de mémoire par rapport au Transformer. Par contre, l'utilisation d'approximations et l'optimisation des blocs de traitement du Reformer lui accordent l'avantage pour la vitesse de traitement, même si l'utilisation approximative du LSH peut nuire à sa précision par rapport au Transformer de base ainsi qu'à notre modèle comme le révèlent nos résultats au prochain chapitre.

	Mémoire	Temps	Temps de Gradient
Transformer	$(b.l.d_{ff} + 3b.n_h.d_q.l^2)n_l$	$(b.l.d_{ff} + 3b.n_h.d_q.l^2)n_l$	$(b.l.d_{ff} + 3b.n_h.d_q.l)n_l$
Reformer	$b.n_h.l.d_k + b.n_h.n_r.l(4l/n_c)^2$	$(b.l.d_{ff} + b.n_h.n_r.l.c)n_l$	$(b.l.d_{ff} + b.n_h.n_r.l.c)n_l$
Notre Modèle	$(b.l.d_{ff} + 2b.n_h.d_q.l^2)n_l$	$(b.l.d_{ff} + 2b.n_h.d_q.l^2)n_l$	$(b.l.d_{ff} + 2b.n_h.d_q.l)n_l$

Tableau 3.2: La complexité totale du calcul spatial, temporel et gradient de notre modèle par rapport au Transformer et au Reformer.

3.3 La génération de dialogues longs

Afin d'améliorer le résultat du dialogue, nous avons introduit deux modèles distincts. Le premier appelé *Cascade* utilise une architecture à un seul encodeur et plusieurs décodeurs (Belainine *et al.*, 2022a). Le second nommé MDA Transformer (*Multi-dimensionnel attention Transformer* (Belainine *et al.*, 2022b)) est une généralisation du modèle présenté à la section précédente.

Les architectures développées sont basées sur la même formulation de problème et le même modèle de dialogue, et leurs entraînements sont fait avec les mêmes ensembles des données, après le même prétraitement syntaxique et la même représentation sémantique présentés dans les sections qui suivent.

3.3.1 Modèle de dialogue

Comme pour le modèle HRED (Sordoni *et al.*, 2015a; Serban *et al.*, 2016b), nous considérons un dialogue comme une séquence $d = (E_1, \cdots, E_M)$ de M énoncés, chacun étant lui-même une séquence de N_m tokens — i.e., $E_m = (w_{m,1}, \cdots, w_{m,N_m})$ —, où chaque token $w_{m,i}$ représente un mot d'un vocabulaire discret $\mathbb{V}$ ou un élément d'analyse non voisé tel un signe de ponctuation. Ensuite, un modèle génératif de dialogue est défini par une distribution de probabilité P de paramètres θ :

$$P_\theta(d) = P_\theta(E_m, \cdots, E_1) = \prod_{m=1}^{M} P_\theta(E_m|E_{<m}) = \prod_{m=1}^{M} P_\theta(w_{m,1}, \cdots, w_{m,N}|E_{<m}) \quad (3.7)$$

Où $E_{<m} = (E_{m-1}, \cdots, E_1)$ est l'historique des énoncés avant E_m. Nous imposons également un vecteur de contexte $V_{m,n}$ pour la position du jeton n. Ce vecteur représente le contexte accumulé de l'historique de la conversation comme présenté dans (Sutskever *et al.*, 2014) :

$$P_\theta(w_{m,n}, w_{m,<n}|E_{<m}) = P_\theta(w_{m,n}, w_{m,<n}|V_{m,n}) = \prod_{n=1}^{N_m} P_\theta(w_{m,n}|V_{m,n}, w_{m,<n}) \tag{3.8}$$

Où $w_{m,<n} = (w_{m,1}, \cdots, w_{m,n-1})$.

Par conséquent, l'équation 3.7 peut être écrite :

$$P_\theta(d) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} P_\theta(w_{m,n}|V_{m,n}, w_{m,<n}) \tag{3.9}$$

La probabilité à droite dans l'équation 3.7 peut être implémentée par un RNR d'états cachés $h_n = f(h_{n-1}, w_n)$, où n représente une position de mot dans l'énoncé m, et f est une fonction non linéaire paramétrée (Mikolov *et al.*, 2010). Quant à l'équation dans son ensemble, elle peut être mise en œuvre par une architecture encodeur-décodeur hiérarchique. Dans le modèle HRED, ceci est accompli par deux encodeurs RNR consécutifs qui traitant respectivement les énoncés d'entrée au niveau des mots et de l'énoncé, et un décodeur RNR utilise la sortie du deuxième encodeur afin de prédire la réponse mot-par-mot (Sordoni *et al.*, 2015a; Serban *et al.*, 2016b).

Les différents paramètres de ce modèle peuvent être appris par l'algorithme de descente du gradient, en utilisant la perte d'entropie croisée (Sukhbaatar et Fergus, 2015) et chaque symbole dans l'énoncé E_m peut être prédit en maximisant l'équation 3.8 (Gao *et al.*, 2018b) :

$$\begin{aligned} \hat{w}_{m,n} &= \underset{w_{m,n}}{Argmax}(\prod_{n=1}^{N_m} p(w_{m,n}|V_m, w_{m,<n})) \\ &= \underset{w_{m,n}}{Argmax}(log \prod_{n=1}^{N_m} p(w_{m,n}|V_m, w_{m,<n})) \\ &= \underset{w_{m,n}}{Argmax}(\sum_{n=1}^{N_m} log \;\; p(w_{m,n}|V_m, w_{m,<n})) \end{aligned} \tag{3.10}$$

Comme les réseaux de neurones préfèrent l'optimisation de fonctions par minimisation au lieu de maximisation, à cause des algorithmes de propagation du gradient. l'équation à contrainte précédente peut être écrite :

$$\hat{w}_{m,n} = \underset{w_{m,n}}{Argmin}(-\sum_{n=1}^{N_m} log \;\; p(w_{m,n}|V_m, w_{m,<n})) \tag{3.11}$$

Tous les modèles de l'état-de-l'art utilisent un vecteur V_m. Les seules différences entre modèles consistent en la manière de construire ce vecteur. On peu citer le modèle HRED (Serban *et al.*, 2016b) qui utilise le dernier vecteur caché du deuxième encodeur, le modèle HRAN qui utilise un vecteur d'attention tiré du deuxième encodeur, les modèles VHRED (Serban *et al.*, 2017c) et VHCR (Zhao *et al.*, 2017) qui utlisent les vecteurs de sortie des auto-encodeurs, et le modèle ReCoSa (Zhang *et al.*, 2019a) qui utilise les vecteurs d'auto-attention.

Dans ce suit, nous adoptons la même logique fondatrice du modèle de dialogue pour construire les deux architectures proposées, Cascade et MDA.

3.4 Le modèle en CASCADE

Dans cette nouvelle approche de type encodeur-décodeur, l'encodeur représente un seul RNR et le décodeur est une séquence de RNRs qui construisent progressivement un contexte de prédiction en se fiant les uns aux autres et en gérant chacun une position de mot dans les énoncés du dialogue. Le contexte final est ainsi basé autant sur les mots que les énoncés précédant la réponse à prédire (Belainine *et al.*, 2022a).

3.4.1 Architecture en Cascade

La figure 3.7 montre le graphe de calcul de l'architecture Cascade. Il est fondé sur quatre GRU formant une séquence composée d'un encodeur suivi de trois décodeurs en

cascade. Le système est entraîné avec des dialogues à quatre tours, $d = (E_1, E_2, E_3, E_4)$, où E_1 et E_3 sont des requêtes de l'utilisateur et E_2 et E_4 sont les réponses du système. L'encodeur A prend E_1 comme entrée pour créer un contexte initial, et chacun des décodeurs B, C et D apprend à prédire l'un des énoncés restants à l'aide d'un contexte agrégé de tous les tours précédents. Ensuite, pendant la phase d'inférence, le système formé reçoit $\{E_1, E_2, E_3\}$ pour prédire E_4.

Pour la prédiction, un processus d'attention multidimensionnelle est utilisé au niveau des décodeurs. Une première attention combine E_2 et le contexte de E_1 – les états cachés de l'encodeur A – afin de l'utiliser dans le décodeur B. Ensuite, une seconde attention combine E_3 avec le contexte de E_1 et les états cachés du décodeur B pour créer un contexte pour le décodeur C. Enfin, une troisième attention combine E_4 avec les contextes des décodeurs B et C, et les états cachés du décodeur C, à utiliser par le décodeur D. Les énoncés réels et prévus (indiqués par un $\hat{E}_2$ sur E) sont alternés pendant l'entraînement, en utilisant un rapport de 1 à 9. Le processus d'attention multidimensionnelle est détaillé plus bas.

Les paramètres du modèle incluent ceux des GRUs de l'encodeur et du décodeur, et du vecteur de contexte du décodeur D dépendent des contextes de l'encodeur A et des décodeurs B et C. Ceux-ci sont appris en maximisant la fonction de log-vraisemblance de l'équation 3.7, comme fait dans le modèle HRED (Sordoni *et al.*, 2015a) :

$$\begin{aligned} \mathscr{L}(S) &= \sum_{m=1}^{M} \log P(E_m | E_{<m}) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N_M} \log P(w_{m,n} | w_{m,<n}, E_{<m}) \end{aligned} \tag{3.12}$$

Le modèle Cascade est entraîné sur des dialogues à 4 tours. Il peut déduire le quatrième énoncé basé sur les trois précédents. Une taille de boîte de dialogue régulière est requise car le modèle apprend des paires requête-réponse. Lorsqu'un dialogue contient moins

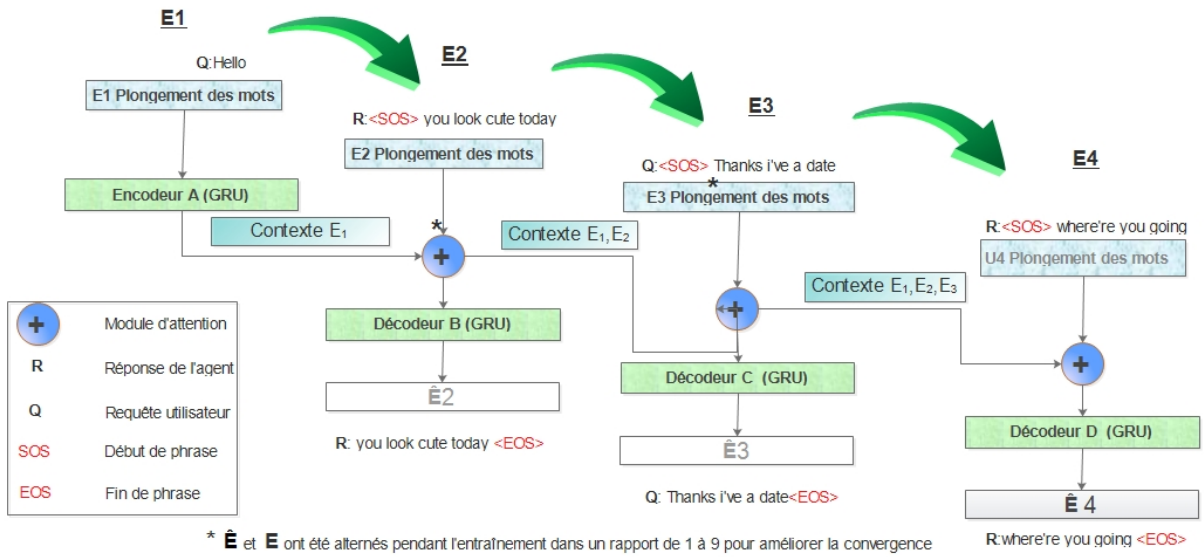


Figure 3.4: Diagramme de flux du modèle Cascade pour la génération de réponse à l'aide de boîtes de dialogue à 4 tours.

de quatre énoncés, il est complété par des énoncés vides.

Durant une passe de l'apprentissage, le modèle calcule les encodages de chacun des trois derniers énoncés et les combine avec les sorties de l'encodeur et des décodeurs associés pour créer le contexte agrégé à l'intention du quatrième décodeur. Lors du passage en arrière, les gradients de l'erreur sont calculés et les paramètres sont mis à jour pour chaque décodeur dans l'ordre inverse.

Comme mentionné précédemment, cette approche peut être vue comme un système dont chaque décodeur apprend par récursion, avec le premier qui apprend la distribution $P(E_2|E_1)$, le second qui apprend la distribution $P(E_3|E_2,E_1)$, et le troisième qui apprend la distribution $P(E_4|E_3,E_2,E_1)$. Cette récursion peut jouer un rôle dans l'évolution de la conversation.

3.4.2 Attention Multidimensionnelle

La figure 3.7 montre que chaque décodeur de Cascade prend deux entrées, l'énoncé actuel et une attention dérivée d'un contexte conjoint construit avec les tours précédents. Ainsi, le décodeur *B* voit le contexte créé par l'encodeur *A*, le décodeur *C* voit les contextes de l'encodeur *A* et du décodeur *B*, et le décodeur *D* voit les contextes de l'encodeur *A* et les décodeurs *B* et *C*. Par conséquent, un contexte global est construit au fur et à mesure que le traitement passe de l'encodeur *A* au décodeur *D*, avec toutes les requêtes et réponses précédentes prises en compte à chaque étape. Plus précisément, le contexte du décodeur *m* pour la position de sortie *n* est déterminé à partir de la séquence d'états cachés de l'encodeur ou du décodeur précédent, et la séquence des contextes précédents. Cela peut être vu comme une extension du mécanisme d'attention proposé par Luong *et al.* (2015a), où seul le contexte actuel contribue à la prédiction cible.

La figure 4.1 illustre les différentes étapes impliquées dans le calcul de l'attention multidimensionnelle proposée. À partir de la séquence d'états cachés de l'encodeur A, $\overline{h_0} = [\overrightarrow{h}_{0,1}; \cdots; \overrightarrow{h}_{0,s}]$, où s est la longueur de la séquence d'entrée, et un contexte pour la position n du décodeur B donné par $\overrightarrow{c}_{0,n} = \sum_{j=1}^{s} \alpha_{n,j} \overrightarrow{h}_{0,j}$, où $\alpha_{n,j} = Softmax\left(\overrightarrow{h}^{T}_{1,n} \overrightarrow{h}_{0,j}\right)$. Les opérations suivantes sont effectuées pour chaque décodeur m et une position de sortie n :

1. Établir le score d'alignement de la position n avec chaque symbole d'entrée en comparant $\overrightarrow{h}_{m,n}$, l'état caché du décodeur m à la position n avec chaque élément de $\overline{h}_{m-1} = [\overrightarrow{h}_{m-1,1}; \cdots; \overrightarrow{h}_{m-1,s}]$ et tous les contextes précédents $\overline{c}_{<m} = [\overline{c}_{m-1}; \cdots; \overline{c}_i; \cdots; \overline{c}_0]$ où un contexte d'un décodeur i est défini par $\overline{c}_i = [\overrightarrow{c}_{i,1}; \cdots; \overrightarrow{c}_{i,s}]$:

$$Score\left(\overrightarrow{h}_{m,n}, \overline{h}_{m-1}, \overline{c}_{<m}\right) = \overrightarrow{h}^{T}_{m,n} \left[\overline{h}_{m-1}; \overline{c}_{<m}\right] \tag{3.13}$$

2. Convertir le résultat en une distribution relative de l'importance de chaque élé-

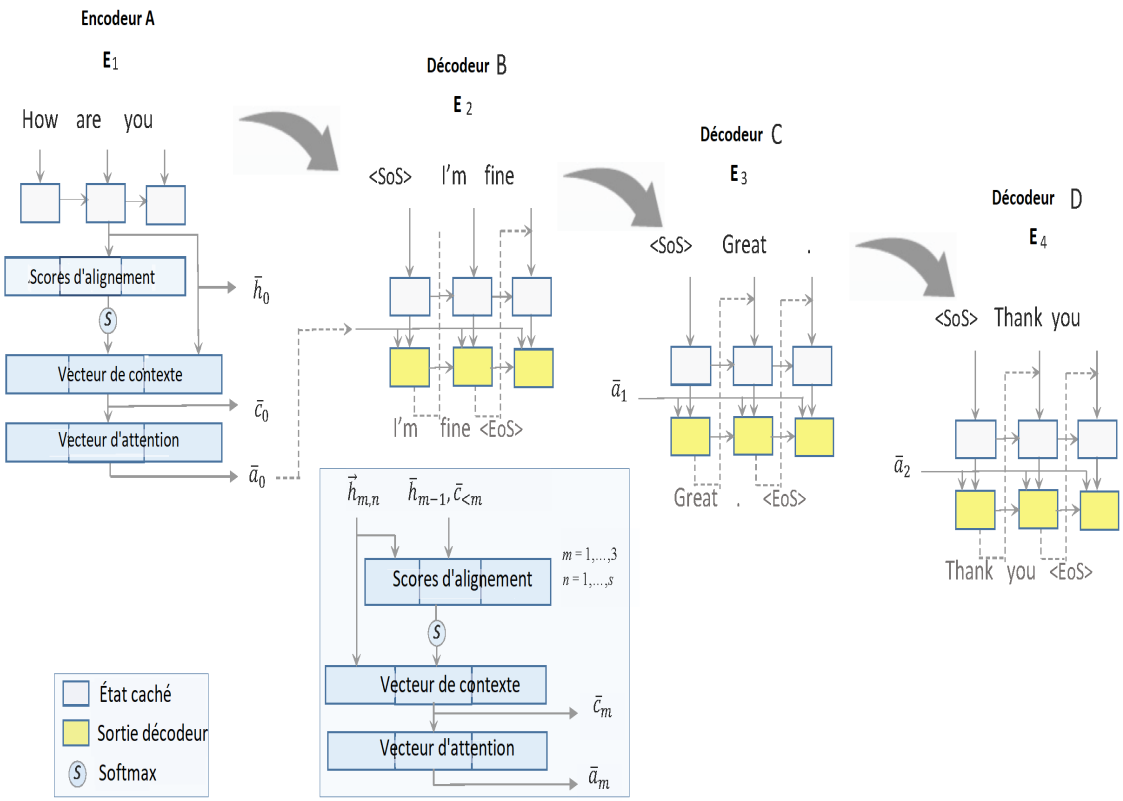


Figure 3.5: Illustration du mécanisme d'attention multidimensionnelle proposé pour l'architecture Cascade.

ment de $\overline{h}_{m-1}$ et de $\overline{c}_{<m}$ dans la prévision de la sortie n :

$$\overrightarrow{\alpha}_n = Softmax\left(Score(\overrightarrow{h}_{m,n}, \overline{h}_{m-1}, \overline{c}_{<m})\right) \tag{3.14}$$

3. Calculer le vecteur de contexte pour prédire la sortie n comme la moyenne pondérée par le score des états sources dans $\overline{h}_{m-1}$ et les contextes précédents $\overline{c}_{<m}$:

$$\overrightarrow{c}_{m,n} = \overrightarrow{\alpha}_n^T [\overline{h}_{m-1}; \overline{c}_{<m}] \tag{3.15}$$

4. Dériver le vecteur d'attention pour le décodeur à partir du vecteur de contexte obtenu et de l'état caché actuel par un réseau à réaction avec une couche cachée (Luong *et al.*, 2015b) :

$$a_{m,n} = f(c_{m,n}, h_{m,n}) = tanh(W_c[c_{m,n}; h_{m,n}]) \tag{3.16}$$

Où W_c est la matrice des coefficients neuronaux et *tanh* est la fonction d'activation de sortie.

3.4.3 Complexité du modèle

Type de couche	Complexité par couche	Opérations séquentielles
Auto-attention	$O(n^2.d_{modèle})$	$O(1)$
Récurrent	$O(n.d^2_{modèle})$	$O(n)$
Récurrent hiérarchique	$O(n.d^2_{modèle}.l_u)$	$O(n.l_u)$
Cascade	$O(n.d^2_{modèle}.l_u)$	$O(n.l_u.n_{dc})$

Tableau 3.3: Complexité par couche, nombre minimum d'opérations séquentielles pour différents types de couches (n est la longueur de la séquence, d la dimension de la représentation, l_u le nombre de l'énoncé, $.n_{dc}$ le nombre de décodeurs.

Le tableau 3.3 montre une comparaison en termes de complexité pour les modèles hiérarchiques de type HRED et dérivés, du Transformer et de Cascade.

Comme expliqué dans Vaswani *et al.* (2017), une couche d'auto-attention implique $O(n^2 d_{modèle})$ opérations par énoncé, car chaque vecteur de mots de taille $d_{modèle}$ doit être comparé à tous les autres vecteurs de mots ; tandis qu'une couche récurrente a une complexité de $O(n d^2_{modèle})$. Cela donne un avantage à l'auto-attention et par conséquent au Transformer, puisque la longueur de séquence n est généralement plus petite que la dimension de la représentation du vecteur de plongement $d_{modèle}$. La séquence des opérations est effectuée en $O(1)$ pour une couche d'auto-attention par opposition à $O(n)$ pour une couche récurrente, donnant ainsi l'avantage au traitement parallèle en faveur du Transformer. Ces avantages demeurent valides dans le multi-tours du Transformer en ajoutant à la complexité la longueur de la séquence des énoncés l_u. Par conséquent, Cascade augmente la complexité en ajoutant le nombre de décodeurs l_{dc} dans le calcul, car les opérations séquentielles sont répétées pour chaque unité de décodeur et pour

tous les décodeurs. Pour cette raison, nous avons limité le nombre des énoncés utilisés à 4.

Les modèles hiérarchiques utilisent un lot d'énoncés d'une taille l_u. Ce lot d'énoncés est exécuté dans le premiers encodeur récurrente qui a une complexité de $O(nd^2_{modèle})$ par énoncé. Par conséquence, la complexité total du modèle est de $O(nd^2_{modèle}.l_u)$.

3.5 Architecture de Transformer à Attention Multidimensionnelle (MDA)

Dans ce modèle, l'encodeur considère l'ensemble de l'historique des dialogues afin de déterminer l'attention de chaque mot d'entrée. Ensuite, le décodeur gère le processus de génération de sortie en fonction des mots d'entrée et de leurs positions, en utilisant l'attention multidimensionnelle pour lier les mots de chaque énoncé de dialogue à la réponse dans un espace bidimensionnel. Les résultats sont empilés au niveau du mot où chaque niveau représente un énoncé afin d'obtenir une attention tridimensionnelle. Ceci est différent du modèle ReCoSa (Zhang *et al.*, 2019a) qui utilise également l'attention de Transformer, mais l'applique au modèle utilisé par HRED où l'ordre d'énonciation est implicitement défini par le premier encodeur RNR, et les énoncés de dialogue sont traités séquentiellement avec les limitations potentielles décrites dans l'état-de-l'art.

Contrairement à l'attention à deux dimensions représentant un alignement entre deux séquences en attribuant un score $a(n,m)$ reliant une position m de l'entrée à une position n de la sortie, l'idée clé de notre travail est un mécanisme d'alignement sur trois types de séquences, en utilisant un score d'attention $a(n,m,k)$ correspondant à trois positions : m pour les entrées, n pour les sorties et k pour la position de l'énoncé dans le dialogue. La figure 3.6 illustre ce concept.

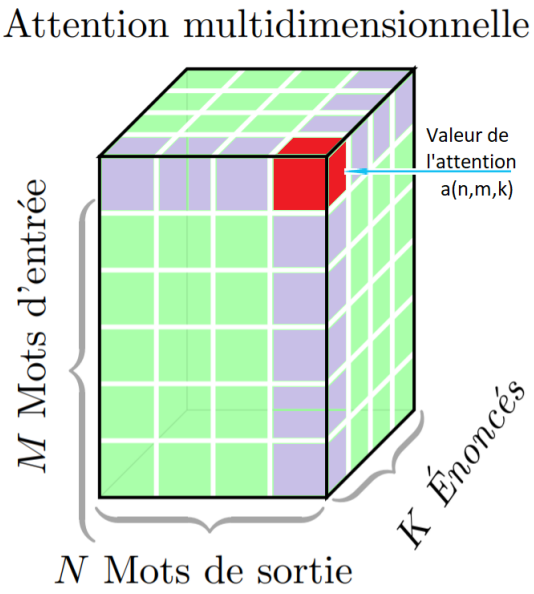


Figure 3.6: Illustration de l'alignement entre trois séquences avec l'attention multidimensionnelle.

3.5.1 Implémentation

La figure 3.7 montre le schéma de notre modèle encodeur-décodeur avec attention multidimensionnelle. Notre modèle est adapté du Transformer [2] (Vaswani *et al.*, 2017) avec un double encodage de position utilisé à l'entrée, un code pour localiser les positions des mots dans les énoncés et l'autre pour les positions des énoncés dans chaque dialogue. Une autre différence avec le Transformer est l'utilisation d'une représentation directe des données de sortie et l'utilisation de similitude cosinus dans la détermination de l'attention au lieu d'un produit scalaire mis à l'échelle de l'approche clé-valeur. Comme déjà présenté à la sous section 3.2.5, ces changements se traduisent par une empreinte moindre , un apprentissage plus rapide et des performances de transduction de séquence améliorées avec une simplicité de la généralisation.

2. https ://github.com/belainine/TransformerMDA

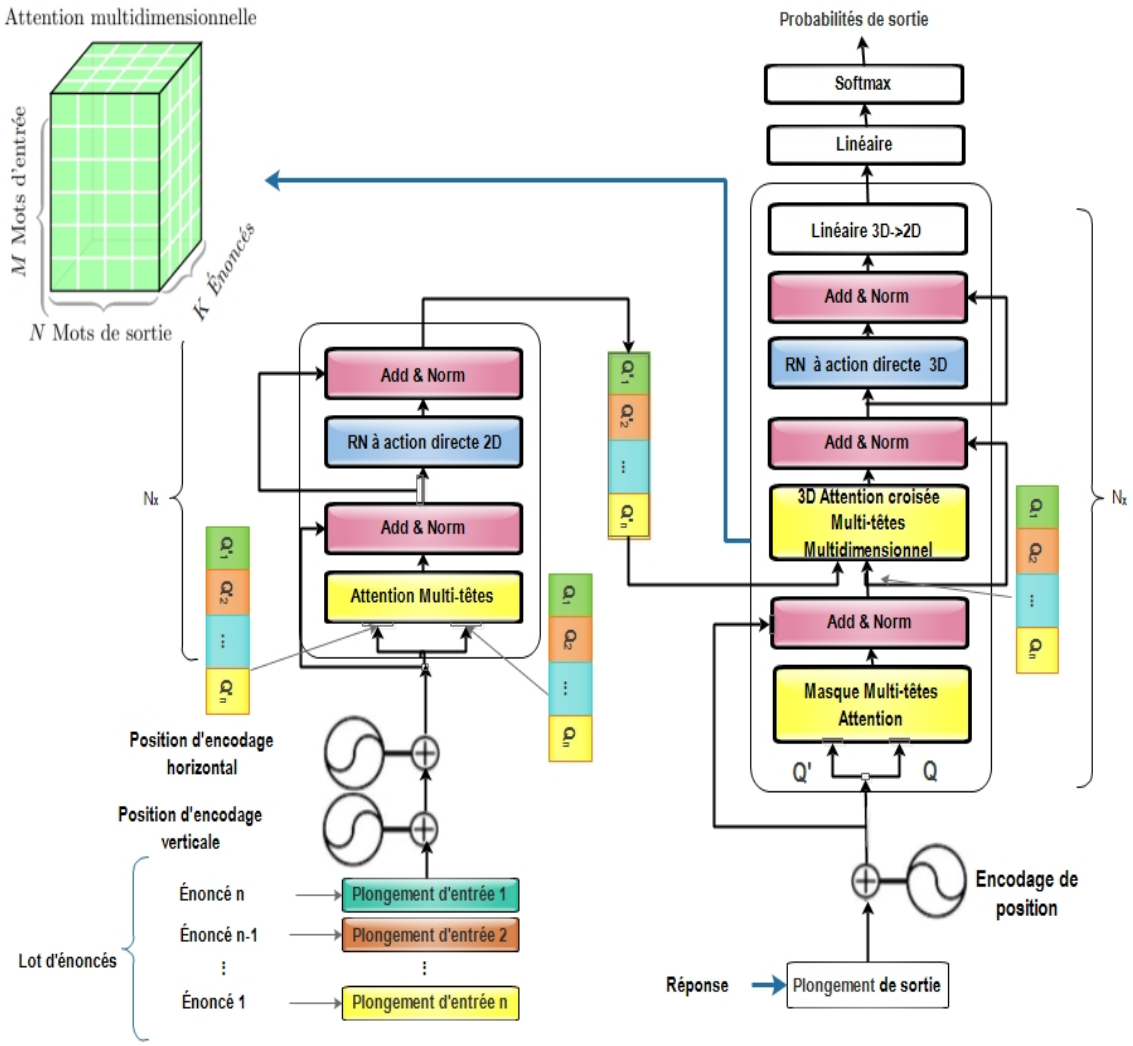


Figure 3.7: Graphe de calcul du modèle Multidimensionnel Transformer pour l'apprentissage de la conversation à l'aide de dialogues créés dans des énoncés.

Basé sur ce qui précède et considérant une séquence de dialogues à n tours $d = (E_1, E_2, \cdots, E_i, \cdots, E_n)$, l'encodeur prend tous les éléments jusqu'à E_n comme entrées et le décodeur prédit E_{n+1} grâce au contexte construit par un tenseur d'attention qui relie les mots d'entrée à la sortie pour chaque énoncé E_i, en utilisant le mécanisme d'attention tridimensionnel qui crée des liens au niveau des mots et de l'énoncé.

Le modèle peut être implémenté pour les dialogues n-tours pour n allant de 1 à $MaxTours$,

et le nombre de mots et éléments syntaxiques allant de 1 à $MaxMots$ par énoncé. Une fois ces deux paramètres définis et le modèle construit, le remplissage par les mots vides est utilisé afin de compléter les dialogues ou les énoncés qui ne satisfont pas à l'un ou à l'autre de ces paramètres. Dans ce travail, $MaxTours$ a été fixé à 10 et $MaxMots$ à 30 à des fins de comparaison de performance avec d'autres modèles de l'état-de-l'art. Par ailleurs, plus de 96 % des phrases traitées pendant la phase de prétraitement pour le grand corpus de Ubuntu (Lowe *et al.*, 2017) étaient de cette longueur ou moins.

3.5.2 Encodeur et décodeur

Nous avons utilisé la même architecture que le Transformer à l'exception du remplacement des matrices Q, K et V de ce dernier par deux matrices Q et $\acute{Q}$, et son encodage de position unidimensionnel par notre encodage à deux positions. Ces changements conduisent à des représentations différentes des données et des mécanismes d'attention comme décrits ci-après :

3.5.3 Encodage de la position des mots et des énoncés

Comme ce modèle n'utilise pas d'unités récurrentes qui fournissent un ordre de mots implicite ; deux vecteurs de position sont ajoutés aux vecteurs de plongement d'entrée pour la localisation horizontale et verticale dans la boîte de dialogue. L'un des vecteurs encode les positions des mots dans les énoncés et l'autre encode les positions des énoncés dans le dialogue.

Étant donné un vecteur de plongement de taille $d_{modèle}$, l'encodage vertical et horizontal des positions des mots dans l'historique du dialogue génère deux vecteurs :

$$PE_{vert}(pos_w) = \begin{cases} PE_{vert}(pos_w)_{2i} = sin(pos_w / 10000^{2i/d_{modèle}}) \\ PE_{vert}(pos_w)_{2i+1} = cos(pos_w / 10000^{2i/d_{modèle}}) \end{cases} \tag{3.17}$$

$$PE_{horiz}(pos_u) = \begin{cases} PE_{horiz}(pos_u)_{2j} = cos(-pos_u/10000^{(2j)/d_{modèle}}) \\ PE_{horiz}(pos_u)_{2j+1} = sin(-pos_u/10000^{(2j)/d_{modèle}}) \end{cases} \quad (3.18)$$

où i est l'indice des éléments. Inverser l'ordre de *sin* et *cos* ainsi que les signes (pos_w et $-pos_u$) dans les deux équations précédentes permet de générer des codes orthogonaux pour les positions des mots et des énoncés, les rendant ainsi indépendants les uns des autres. Ensuite, leur somme fournit un code identifiant simultanément l'ordre de mot et l'ordre d'énonciation dans la boîte de dialogue d'entrée. Cette opération statique peut être effectuée à l'avance et les résultats stockés en mémoire pour un traitement plus rapide.

Une fois les deux vecteurs de position définis, le vecteur de plongement d'un mot situé à la position i d'un énoncé j dans le dialogue devient :

$$Plongement\ Final_{i,j} = Plongement_{i,j} + PE\ vertical_i + PE\ horiontal_j \quad (3.19)$$

Par conséquent, deux mots dans la séquence du dialogue seront intégrés plus étroitement en fonction non seulement de leur proximité sémantique, mais également de leurs positions relatives dans la séquence. Ainsi, chaque mot $w_{i,j}$ fournira l'information sur sa position dans l'énoncé $PE\ vertical_i$ et la position de l'énoncé $PE\ horiontal_j$ dans la conversation de manière explicite. La figure 3.8 illustre l'encodage de position obtenu pour les mêmes positions de mot dans différents énoncés lorsqu'on considère une profondeur de dialogue de 10 énoncés ou une profondeur d'énoncé de 30 mots, et une taille de chaque vecteur de plongement de mot égale à $d_{modèle} = 256$.

La figure 3.9 représente les plongements de position pour tous les pas de temps, générés pour les mêmes positions des mots dans une séquence de 10 énoncés, en supposant une profondeur d'énonciation de 30 mots et une taille de vecteur de mot de plongement de 256. Chaque figure montre le code de position composite obtenu par l'ajout de celui

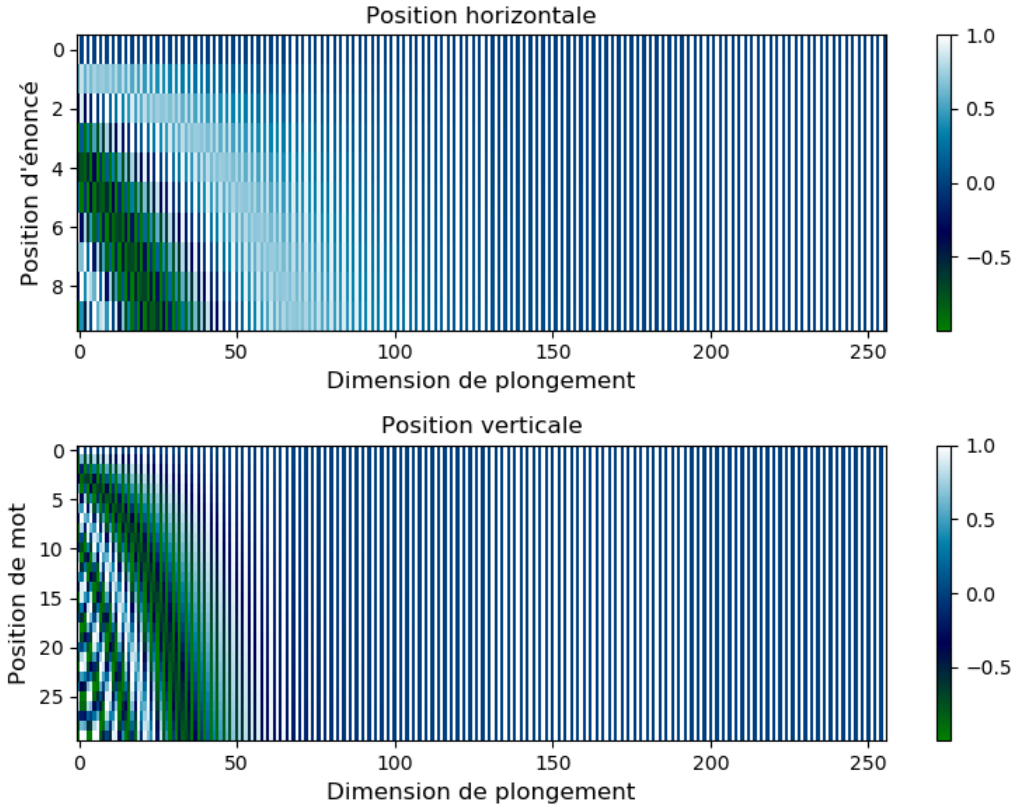


Figure 3.8: Cartes thermiques des codes de position générés pour un dialogue de 10 énoncés ou moins (à gauche) et une taille d'énoncé de 30 mots ou moins (à droite), en utilisant $d_{modèle} = 256$.

de la position d'énoncé à celui des positions du mot. Cela reflète la dégradation de l'intensité entre la position d'un mot et celle de ces voisins, permettant ainsi d'avoir une idée sur la position relative entre deux mots. Une autre propriété du codage de position s'observe dans la distance entre les pas de temps voisins qui est symétrique et diminue bien avec le temps comme nous montre la figure 3.10. La dégradation des couleurs présentée dans la figure 3.10 montre que les codes de positions générés par l'équation 3.19 sont tous distincts.

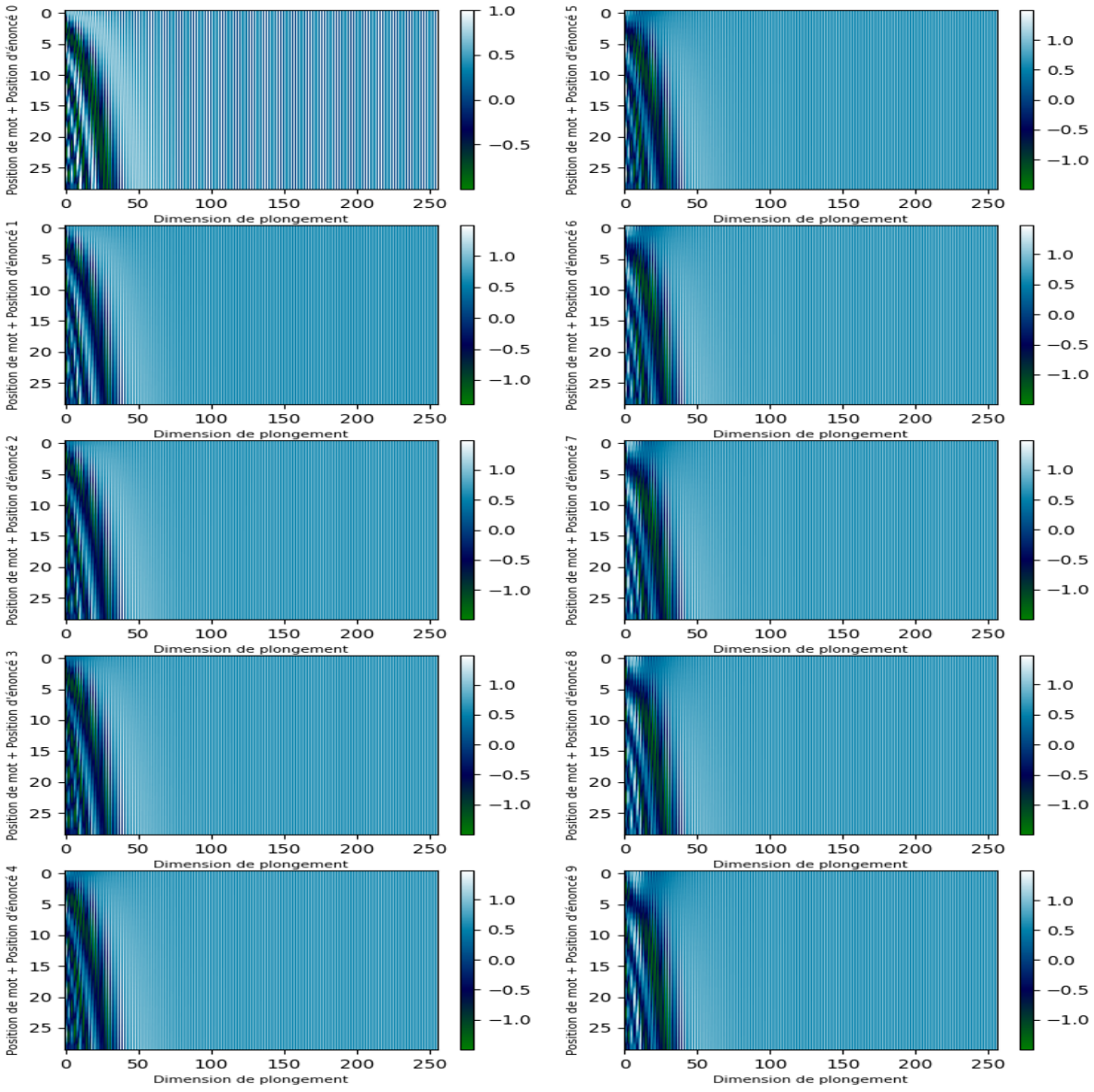


Figure 3.9: Illustration des codes générés pour les mêmes positions de mots dans une séquence de 10 énoncés, en supposant une profondeur d'énonciation de 30 mots et en utilisant $d_{modèle} = 256$. Chaque figure montre le code de position composite obtenu en ajoutant le code de position de l'énoncé au code des position du mot

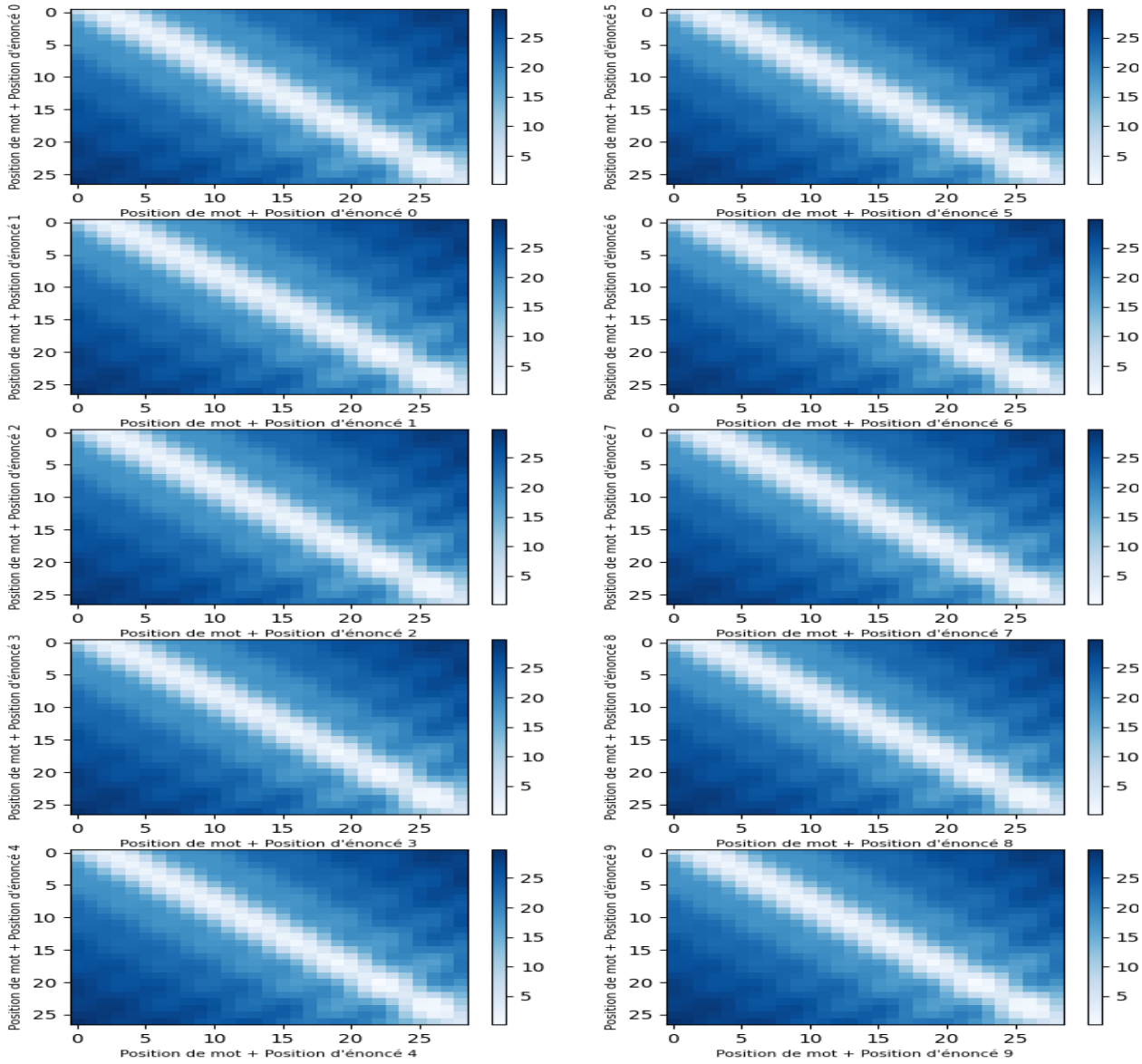


Figure 3.10: Illustration de la distance entre les plongements de position pour tous les pas de temps, en ajoutant le code de position d'énoncé au code des positions du mots.

3.5.4 Attention Multidimensionnelle

Dans l'encodeur, notre modèle projette chaque vecteur d'entrée marqué par son code à deux positions sur deux vecteurs de représentation q et $\acute{q}$, et deux matrices Q et $\acute{Q}$

sont obtenues par l'empilement des entrées des mots de chaque énoncé. Ensuite, la similarité cosinus des deux matrices est calculée avant d'appliquer la fonction $Softmax$ au résultat. Enfin, nous utilisons les résultats pour de pondérer les éléments de $\acute{Q}$:

$$\begin{aligned} Attention_{2D}(Q,\acute{Q}) &= Softmax\left(cos\left(Q\acute{Q}^T\right)\right)\frac{\acute{Q}}{\|\acute{Q}\|} \\ &= Softmax\left(\frac{Q\acute{Q}^T}{\|Q\|\|\acute{Q}^T\|}\right)\frac{\acute{Q}}{\|\acute{Q}\|} \end{aligned} \tag{3.20}$$

L'ensemble du processus est répété pour tous les énoncés d'un dialogue d pour le lot d'entrée et une pile de matrices d'auto-attention est obtenue à la fin. Cette pile identifie chaque dialogue avec sa position.

Le décodeur démarre de la même manière que l'encodeur, avec les mots générés précédemment en entrée et l'utilisation du calcul d'auto-attention masqué afin d'ignorer ceux qui ne le sont pas encore.

Ensuite, l'attention de l'encodeur-décodeur est calculée entre la pile de sortie de l'encodeur et les résultats du décodeur, créant ainsi une attention 3D pour prédire la prochaine sortie de ce dernier. Ainsi, étant donné la pile de matrices $\acute{Q}_i$ à la sortie de l'encodeur, avec $i \in \{1,\cdots,n\}$ et n la taille de la séquence des énoncés, et la matrice Q des mots de la réponse du décodeur, le calcul d'attention suivant est fait :

$$\begin{aligned} \alpha &= Attention_{3D}(Q,\acute{Q}_{i<=n}) \\ &= Softmax_{sortie}\left(\left[cos\left(Q\acute{Q}_1^T\right) \quad \cdots \quad cos\left(Q\acute{Q}_n^T\right)\right]\right)\frac{\acute{Q}_i}{\|\acute{Q}_i\|} \\ &= Softmax_{sortie}\left(\left[\frac{Q\acute{Q}_1^T}{\|Q\|\|\acute{Q}_1^T\|} \quad \cdots \quad \frac{Q\acute{Q}_n^T}{\|Q\|\|\acute{Q}_n^T\|}\right]\right)\frac{\acute{Q}_i}{\|\acute{Q}_i\|} \end{aligned} \tag{3.21}$$

Nous appelons ce mécanisme « *Multi-dimensionnel Attention* » (MDA), car il reflète les dépendances entre les mots de la sortie du décodeur et les mots d'entrée à la fois au

niveau du mot dans les énoncés et au niveau des énoncés dans le dialogue.

L'attention multidimensionnelle fonctionne comme suit : (1) le modèle projette chaque sortie de plongement dans un vecteur $\acute{q}$ et la matrice $\acute{Q}$ est obtenue par l'empilement de ces vecteurs ; (2) la similitude cosinus est calculée via chaque sortie de l'encodeur Q_i avec la réponse $\acute{Q}$ générant ainsi un tenseur 3D. Le tenseur 3D représente la similitude cosinus entre la matrice des mots de la réponse Q et les mots des énoncés encodés $[\acute{Q}_1 \cdots \acute{Q}_i \cdots \acute{Q}_n]$. Le tenseur résultat du produit est de taille (Taille des mots d'entrée ,Taille des mots de sortie, Taille des énoncés).

Afin d'obtenir les poids d'attention $a_{i,j,s}$ qui reflètent la participation de chaque vecteur $q_{i,j,s}$ dans le contexte, une normalisation $Softmax$ est appliquée au tenseur pour chaque mot de sortie, de sorte que la somme des éléments de chaque matrice de dimension (taille des mots d'entrée, indice s , taille des énoncés) soit égale à 1 :

$$a_{i,j,s} = Softmax_{sortie}(z_{i,j,s}) = \frac{e^{z_{i,j,s}}}{\sum_{m=1}^{M}\sum_{n=1}^{N} e^{z_{n,m,s}}} \text{ pour tout } i \in \{1..N\} \text{ et } j \in \{1..M\} \tag{3.22}$$

Où les scores $z_{i,j,s} = cos(q_s, \acute{q}_{i,j})$ pour tout $i \in \{1..N\}$ et $j \in \{1..M\}$; s est l'indice d'une sortie donnée ; N est la taille maximale des mots de l'énoncé ; M est la taille maximale des énoncés d'un dialogue d. Le tenseur obtenu donne les poids relatifs de l'attention, dans le but de nous renseigner sur la participation de chaque mot d'entrée pour générer un mot de sortie. Figure 3.11 illustre le processus.

Afin de calculer les vecteurs des contextes α de l'attention 3D, une pondération est appliquée par le produit du tenseur normalisé et les matrices des entrées $[\acute{Q}_1, \cdots \acute{Q}_i, \cdots, \acute{Q}_n]$.

Cela permet d'obtenir une pondération sur tous les mots des énoncés, dont les vecteurs des mots sont multipliés par les scores faibles et réduits à des vecteurs qui ont des va-

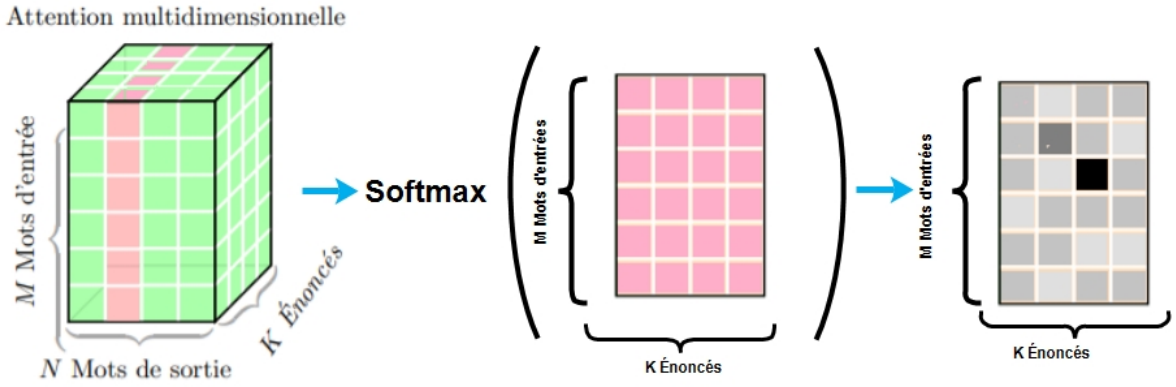


Figure 3.11: Application de la fonction *Softmax* à une matrice extraite de tenseur pour un indice de sortie s et sur tous les mots qui correspondent à tous les énoncés.

leurs faibles ; et les vecteurs des mots sont multipliés par les scores élevés sont augmentés. Le résultat par vecteur $\alpha_{i,s}$ est calculé par l'équation 3.23, tel que $s \in [0, \cdots, N-1]$ est l'indice d'une sortie donnée ; M est la taille maximale des mots de l'énoncé ; $i \in [0, \cdots, K-1]$ est l'indice d'une énoncé d'un dialogue d.

$$\alpha_{i,s} = \sum_{j=1}^{M} a_{i,j,s} \cdot \acute{q}_{i,j} \tag{3.23}$$

Les vecteurs $\alpha_{i,s}$ obtenus, forment un tenseur α en trois dimensions ; la première dimension est la taille de chaque vecteur égale à $d_{modèle}$; le deuxième composant a une taille de sortie de $MaxMots$; le troisième composant représente la taille des énoncés avec la forme totale $(MaxMots, d_{modèle}, MaxTours)$.

Une fois les vecteurs α d'attention 3D calculés, le traitement du décodeur restant se déroule comme dans le modèle original du Transformer, avec les positions de mots et d'énoncés prises en compte. Ainsi, les profondeurs d'entrée et de sortie de chaque composant du réseau à action directe à deux couches de la figure 3.7 sont égales à $d_{modèle} \cdot MaxTours$, avec une profondeur de couche cachée d_{ff} plus élevée que $d_{modèle} \cdot MaxTours$ (Vaswani *et al.*, 2017).

Ensuite, le réseau de projection linéaire transforme le format 3D des données en un format 2D. Ce réseau linéaire utilise les vecteurs mots obtenus et les significations contextualisées des énoncés pour chaque position de mot de sortie, avec la forme totale $(MaxMots, d_{modèle}, MaxTours)$, pour produire une sortie de forme $(MaxMots, d_{modèle})$ compatible avec l'entrée du classificateur linéaire final ou de la couche de décodage suivante.

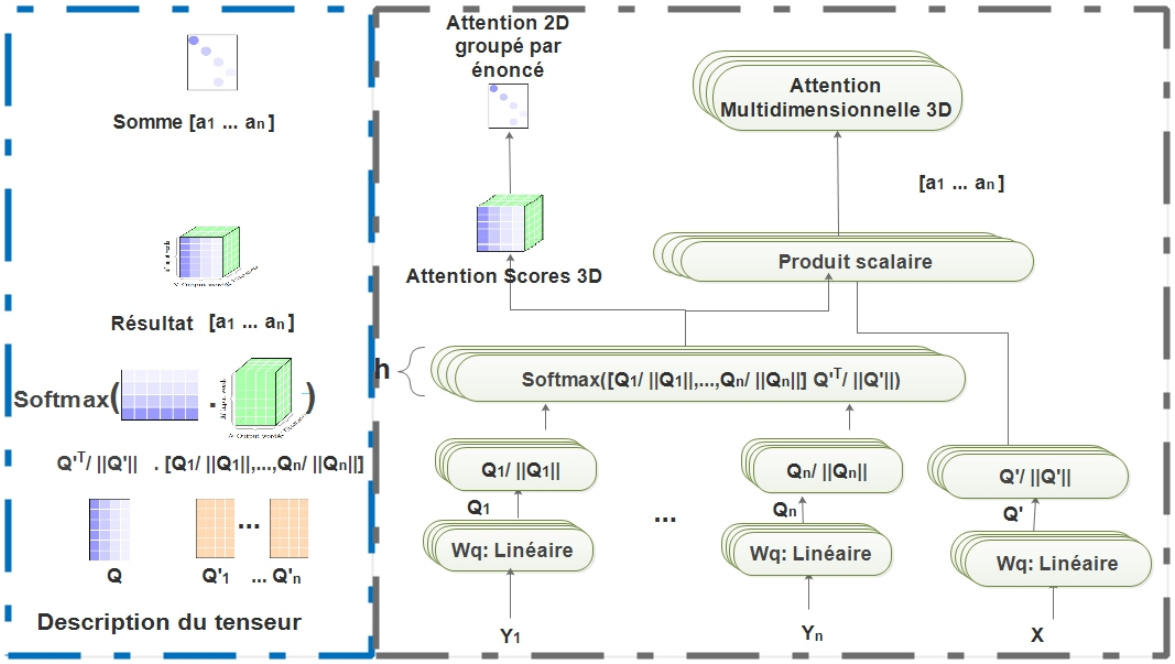


Figure 3.12: Schéma fonctionnel du mécanisme d'attention multidimensionnelle proposé (à droite) avec une illustration graphique (à gauche).

La figure 3.12 résume notre mécanisme d'attention multidimensionnelle. Elle montre deux groupes de matrices apprises W_q et $W_{\acute{q}}$ qui servent à former les vecteurs de représentation q_i et $\acute{q}_i$ à partir des vecteurs d'entrée et de l'encodage de position. Les vecteur produits sont empilés pour créer les matrices Q et $\acute{Q}_i$ avant de calculer l'attention multidimensionnelle. La sommation en haut produit une attention au niveau de l'énoncé et permet d'identifier les énoncés ayant le plus contribué à chaque sortie. Cette étape est optionnelle et ne fait pas partie du processus d'apprentissage.

3.5.5 Complexité du modèle

Type de couche	Complexité par couche	Opérations séquentielles
Auto-attention	$O(n^2.d_{modèle})$	$O(1)$
Récurrent	$O(n.d^2_{modèle})$	$O(n)$
Récurrent hiérarchique	$O(n.d^2_{modèle}.l_u)$	$O(n.l_u)$
Modèle MDA	$O(n^2.d_{modèle}.l_u)$	$O(1)$

Tableau 3.4: Complexité par couche, nombre minimum d'opérations séquentielles pour différents types de couches (n est la longueur de la séquence, d la dimension de la représentation, l_u la taille de l'énoncé.

Comme expliqué dans Vaswani *et al.* (2017) et repris plus haut, une couche d'auto-attention implique $O(n^2 d_{modèle})$ opérations par énoncé, tandis qu'une couche récurrente a une complexité de $O(n d^2_{modèle})$. Cela donne un avantage à l'auto-attention, puisque la longueur de séquence n est généralement plus petite que la dimensionnalité de la représentation de vecteur de plongement $d_{modèle}$. De plus, grâce à la fonctionnalité d'accès direct de l'auto-attention à l'information, la séquence des opérations est effectuée en $O(1)$ par opposition à $O(n)$ pour une couche récurrente, donnant ainsi l'avantage à un traitement parallèle en faveur de l'auto-attention. Ces avantages sont préservés dans l'attention multidimensionnelle proposée, car elle ne fait qu'ajouter l_u, le nombre des énoncés dans le dialogue, à la complexité de calcul. Le tableau 3.4 résume les complexités par couche.

3.5.6 La sortie du décodeur

Une fois l'attention 3D calculée, le traitement du décodeur restant se déroule comme dans le modèle original du Transformer, avec les positions de l'énoncé ajoutées prises

en compte. Ainsi, les profondeurs d'entrée et de sortie de chaque composant du réseau à action directe à deux couches de la figure 3.7 sont égales à $d_{modèle} \cdot MaxTours$. La profondeur de la couche cachée d_{ff} pour ce réseau est beaucoup plus grande que l'entrée. (nous avons utilisé $d_{ff} = 4096$). Ensuite, on utilise un réseau de projection linéaire du mot obtenu et les significations contextualisées de l'énoncé, afin de projeter une forme 3D ($MaxMots, d_{modèle}, MaxTours$) et en produire une sortie 2D ($MaxMots, d_{modèle}$) compatible avec l'entrée du classificateur final ou de la couche de décodage suivante.

La couche linéaire utilisée par la sortie de décodage est la même que celle utilisée par le modèle de base du Transformer (Vaswani *et al.*, 2017). Cette couche linéaire est un simple réseau de neurones entièrement connecté qui projette un vecteur de taille $d_{\text{modèle}}$ produit par la pile des couches de décodeurs dans un vecteur de même taille que le vecteur du vocabulaire. Chaque composante de ce vecteur correspondant au score d'un mot unique. La couche Softmax transforme ensuite ces scores en probabilités. La cellule avec la probabilité la plus élevée est choisie, et le mot qui lui est associé est choisir en sortie.

3.6 Conclusion

Ce chapitre a présenté la méthodologie développée dans ce projet de recherche. Le projet s'est déroulé en 3 phases dont chacune comporte des contributions complémentaires pour réaliser un agent génératif. Le modèle final est basé sur une architecture inspirée du Transformer et doté d'un mécanisme d'attention multidimensionnelle. Les 3 phases sont :

- Une optimisation du modèle de Transformer grâce au changement de son modèle d'attention par une attention basée sur la similitude cosinus.
- La création d'un modèle en cascade doté d'un mécanisme d'attention multidimensionnelle pour les tâches de dialogue multi-tours.
- La généralisation du modèle de Transformer de la première phase par l'introduc-

tion du mécanisme d'attention multidimensionnelle présenté dans la deuxième phase.

Dans le chapitre suivant, nous présentons les processus d'évaluation menés afin d'expérimenter les différents modèles proposés et décrits dans ce chapitre.

CHAPITRE IV

ÉVALUATION DE LA MÉTHODOLOGIE

4.1 Introduction

Ce chapitre présente l'évaluation des modèles présentés au chapitre précédent. Nous avons mené plusieurs expériences en vue d'évaluer les performances de ces modèles et les comparer à l'état-de-l'art. Les évaluations ont porté tant sur la génération de réponses dans les dialogues que sur la traduction automatique neuronale. L'objectif ultime est de montrer la supériorité de notre modèle transformer MDA par rapport à d'autres modèles basés sur l'attention, les RNR ou les deux.

4.2 Cadre expérimental

Afin de valider nos hypothèses de recherche, nous avons proposé un protocole expérimental qui inclut la comparaison avec des modèles de référence et utilise trois corpus différents.

- Modèles de référence : Nous avons comparé nos modèles Cascade et transformer MDA à 6 modèles de l'état de l'art :
    - ⋆ Le transformer (Vaswani *et al.*, 2017) est utilisé comme référence pour évaluer l'effet d'utiliser seulement deux matrices de représentations et la similarité cosinus dans le calcul de l'attention, dans le but de montrer que l'amélioration de prédiction obtenue provient de nos modifications à cette

architecture.

- ⋆ L'encodeur-décodeur récurrent hiérarchique (HRED [1]) (Serban *et al.*, 2016b), L'encodeur-décodeur variationnel récurrent hiérarchique (VHRED) (Serban *et al.*, 2017c) et les RNRs de conversation hiérarchique variationnelle (VHCR) (Zhao *et al.*, 2017) servent à valider l'usage d'une attention multi-dimensionnelle.
- ⋆ Le réseau hiérarchique d'attention récurrente pour la génération de réponses (HRAN) (Xing *et al.*, 2018) et les contextes pertinents avec auto-attention (ReCoSa) (Zhang *et al.*, 2019a)) servent à valider l'usage du transformer MDA par rapport aux modèles hiérarchiques qui essayent de combiner attention et RNR pour la génération de dialogues.

Les comparaisons des différents modèles reposent sur les mêmes métriques de performance décrites plus loin.

- Corpus de données : Deux types sont utilisés, l'un pour évaluer les modèles de dialogue et l'autre pour évaluer les modèles de traduction.

  1. Corpus de dialogue : les modèles ont été entraînés avec les ensembles de données suivants :

     (a) Le corpus de dialogues de films Cornell (Cornell Movie) (Banchs, 2012) qui contient 220 579 conversations de 617 films.

     (b) Le corpus de dialogue Ubuntu (Lowe *et al.*, 2017) qui contient 930 000 conversations en texte brut des canaux Ubuntu IRC (Internet Relay Chat).

     (c) Le DailyDialog (Li *et al.*, 2017b) qui contient 13 118 conversations de la vie quotidienne pour échanger des informations et renforcer les liens

1. https ://github.com/natashamjaques/AHierarchicalLatent-StructureforVariationalConversation-Modeling

sociaux.

Les trois corpus ont été choisis pour leurs différents sujets et tailles, afin de créer un cadre de conversation ouvert raisonnable.

Nous avons limité la taille des énoncés de chacun à 30 mots et tronqué ceux qui dépassent cette limite. Le réglage de la taille d'entrée de l'encodeur et des décodeurs à cette valeur a permis une bonne efficacité de calcul. Tous les ensembles de données ont été choisis en anglais en raison de leurs disponibilités ainsi que celles des ressources de prétraitement et la détection des entités nommées. Les données de dialogue sont bruitées ; elles proviennent des réseaux sociaux. Pour cette raison, nous avons utilisé l'outil *spacy* qui offre des modèles [2] entraînés sur les dialogues des médias sociaux disponibles en anglais. En outre, ces corpus se retrouvent souvent utilisés dans la littérature (Li *et al.*, 2017b; Zhang *et al.*, 2019a; Xing *et al.*, 2018; Serban *et al.*, 2016b). Le tableau 4.1 nous montre des exemples des dialogues dans des trois corpus utilisés.

Corpus	**Dialog**
DailyDialog	Is Tom available please ? → He's on the other line , can you hold for a minute ? → Sure . → He'll be right with you .
Cornell Movie	Hey, sweet cheeks.→Hi, Joey.→ You're concentrating awfully hard considering it's gym class.
Ubuntu	does anyone know if future versions of ubuntu will give you an option of installing lilo ? → I like lilo → been using it forever

Tableau 4.1: Exemples de dialogues dans les trois corpus utilisés.

2. Corpus de traduction : Les corpus de traduction sont des corpus parallèles où l'on retrouve des phrases alignées par paires, les unes sont rédigées dans

2. https ://spacy.io/models/en

une langue source et les autres dans une langue cible. Le tableau 4.2 donne des exemples. L'édition 2014 de l'atelier sur la traduction machine WMT-2014 [3] a rendu disponibles des corpus parallèles pour plusieurs paires de langues, dont l'anglais vers l'allemand et l'anglais vers le français. Ces corpus sont extraits des documents d'institutions internationales comme le Parlement Européen, les Nations Unis ou encore de sites d'échanges multilingues.

(a) Corpus WMT-2014 anglais-allemand : il a été construit à partir de l'ensemble des travaux du parlement européen [4](Koehn, 2005), de commentaires d'actualités en ligne [5](Smith *et al.*, 2013), et du corpus *Common Crawl* [6] qui a été collecté à partir de pages Web rédigées dans différentes langues (Bojar *et al.*, 2014). L'ensemble des ces éléments représente 4,5 millions (4 508 785) de paires de phrases.

(b) Corpus WMT-2014 anglais-français : ce corpus est aussi dérivé des trois ensembles de données précédents, en y ajoutant le corpus d'extraits en ligne *French-English Gigaword* corpus [7] (Callison-Burch *et al.*, 2011) et le corpus parallèle des Nations Unies [8] (Ziemski *et al.*, 2016). L'ensemble représente 40 millions (40 836 876) de paires de phrases.

Les deux ensembles d'entraînement précédents ont été choisis pour la comparaison avec les modèles de l'état de l'art mentionnés plus haut en utilisant

3. http ://www.statmt.org/wmt14

4. http ://www.statmt.org/wmt14/training-parallel-europarl-v7.tgz

5. http ://www.statmt.org/wmt14/training-parallel-nc-v9.tgz

6. http ://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz

7. http ://www.statmt.org/wmt10/training-giga-fren.tar

8. http ://www.statmt.org/wmt13/training-parallel-un.tgz

les mêmes conditions et données. Cela s'applique notamment au transformer original de Vaswani *et al.* (2017).

Corpus	Texte en langue source	Texte en langue cible
anglais-allemand	You will be aware from the press and television that there have been a number of bomb explosions and killings in Sri Lanka .	Wie Sie sicher aus der Presse und dem Fernsehen wissen , gab es in Sri Lanka mehrere Bombenexplosionen mit zahlreichen Toten .
anglais-français	madam president , i should like to draw your attention to a case in which this parliament has consistently shown an interest .	madame la présidente , je voudrais attirer votre attention sur un cas dont s'est régulièrement occupé le parlement .

Tableau 4.2: Exemples des paires de phrases dans les corpus d'entraînement pour la traduction.

4.2.1 Processus d'évaluation et métriques

La validation des solutions proposées sont faits en 2 étapes : 1) validation pour la traduction machine ; 2) validation pour la génération de réponses dans les dialogues. Les modèles sont comme suit :

- ⋆ Modèle de traduction basé sur transformer (Vaswani *et al.*, 2017) : Ce modèle s'adresse aux problèmes de transduction séquence vers séquence. Les métriques de performance correspondantes sont le score BLEU (Papineni *et al.*, 2002), la perplexité (Sennrich, 2012), le taux d'erreur de mots (Popović et Ney, 2007) et la précision (*Accuracy correct word-ACW*) (Streiner et Norman, 2006).
- ⋆ Modèle de génération de dialogues : deux types d'évaluations sont utilisés comme discuté au chapitre 2 :
  1. L'évaluation automatique : elle utilise la mesure BLEU comme mesure de chevauchement, la perplexité pour évaluer les modèles de langue et les métriques basées sur les plongements de mots proposées par (Liu *et al.*, 2016). Ces dernières comprennent la moyenne des plongements (Mitchell et La-

pata, 2008), les vecteurs d'inclusion extrême (Forgues *et al.*, 2014) et la correspondance gourmande (Mitchell et Lapata, 2008; Rus et Lintean, 2012a).

2. L'évaluation manuelle : cinq utilisateurs évaluent à l'aveuglette la performance des différents modèles en termes de cohérence des phrases générées. Pour chaque modèle et corpus, 100 boîtes de dialogue sont choisies au hasard et chaque évaluateur est invité à classer les réponses sans connaître le modèle générateur. Ensuite, les gains, pertes et égalités des modèles sont calculés à partir des classements faits par les cinq évaluateurs pour chaque corpus et modèle. Cette méthode d'évaluation a déjà été appliquée dans des modèles de l'état de l'art (Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Park *et al.*, 2018; Xing *et al.*, 2018; Zhang *et al.*, 2019a)

4.3 Évaluation du modèle Transformer avec représentation DSM et attention basée sur la similarité cosinus

Afin de valider notre méthode d'optimisation du modèle transformer DSM [9], une comparaison est faite avec le modèle original. Nous avons téléchargé une implémentation en *pytorch* [10] de ce dernier en *pytorch* [11] et y avons intégré notre module d'attention sans toucher au reste du modèle. Nous avons également utilisé les mêmes hyperparamètres que ceux du document principal de transformer (Vaswani *et al.*, 2017), à l'exception de la taille des lots traités qui a été réduite de 1024 à 120 en raison de nos ressources de calcul limitées. Il en va de même pour les données d'entraînement, de développement et de test extraites du site de la campagne d'évaluation WMT 2014 [12]. Nous nous assurons

9. https ://github.com/belainine/TransformerDSM

10. https ://pytorch.org/

11. https ://github.com/jadore801120/attention-is-all-you-need-pytorch

12. http ://www.statmt.org/wmt14

ainsi que toute différence de performance entre les deux modèles serait uniquement due à celles que nous avons apportées au modèle original.

Nos expérimentations comprenaient également des comparaisons avec le modèle Reformer (Kitaev *et al.*, 2020). Ceci a été réalisé en utilisant le module Encoder-Décoder du Reformer-pytorch version 1.1.4 [13].

4.3.1 Données d'entraînement

Nous avons utilisé les tâches de traduction automatique *WMT 2014*[27] anglais-allemand et anglais-français (Bojar *et al.*, 2014). L'ensemble de données anglais-allemand [14] de 2014 comprenant environ 4,5 millions paires de phrases pour des vocabulaires source et cible partagent près de 37 000 mots.

L'ensemble de données anglais-français[26] de 2014 se composant d'environ 40 millions paires de phrases pour des vocabulaires source et cible partagent près de 32 000 mots (Zhou *et al.*, 2016). Nous avons utilisé le Parseur Moses pour diviser les différentes phrases [15].

Les phrases d'entrée et de sortie ont utilisé une longueur d'une séquence maximale de 120. Enfin, chacun des ensembles de développement [16] et de test [17] des deux tâches de traduction comprenait 3 003 phrases.

L'entraînement a été effectué sur une seule carte *GPU NVIDIA GTX1080 TI*, en uti-

---

13. https ://pypi.org/project/reformer-pytorch/

14. https ://www.tensorflow.org/datasets/catalog/wmt14_translate

15. https ://www.nltk.org/_modules/nltk/tokenize/moses.html

16. http ://www.statmt.org/wmt14/dev.tgz

17. http ://www.statmt.org/wmt14/test-filtered.tgz

lisant les mêmes hyperparamètres du document Transformer (Vaswani *et al.*, 2017), à l'exception de la taille du lot qui a été réduite de 1024 à 120 phrases en raison de la limite de mémoire de 11 Go de la carte. Cela ne devrait pas avoir d'impact sur nos comparaisons. En effet, la même implémentation est exécutée pour les deux modèles en dehors de la projection des données et des calculs d'attention. La taille utilisée dans l'algorithme de la recherche en faisceau (*Beam search* (Meister *et al.*, 2020)) pour tous les modèles est égale à 4.

4.3.2 Évaluations et métriques

Nous avons complété diverses évaluations à la fois en se basant sur le temps d'apprentissage et les performances de la traduction. Le temps d'apprentissage a été mesuré en heures, en nombre d'itérations et en nombre d'étapes par les itérations. Les performances de la traduction ont été évaluées avec la perplexité (*PPL*) (Sennrich, 2012) et le score *BLEU*, semblable à l'étude sur le transformer (Vaswani *et al.*, 2017). Nous avons aussi ajouté le taux des mots corrects (ACW) (Streiner et Norman, 2006) ainsi que le taux d'erreurs des mots (WER), qui est un dérivé de la distance de *Levenshtein* (Popović et Ney, 2007). Les scores *BLEU* ont été déterminés avec *SacreBLEU* [18].

4.3.3 Résultats

La figure 4.3 montre l'évolution temporelle de la meilleure convergence de la perplexité (*PPL*). Cette convergence est appliquée au corpus d'évaluation de la tâche de traduction automatique anglais-français. La comparaison est basée sur le modèle proposé à base du cosinus, le transformer de base (Vaswani *et al.*, 2017) ainsi que le Reformer (Kitaev *et al.*, 2020).

La table 4.3 fournit les statistiques d'entraînement lors de l'utilisation du nombre de

18. https ://github.com/mjpost/sacrebleu

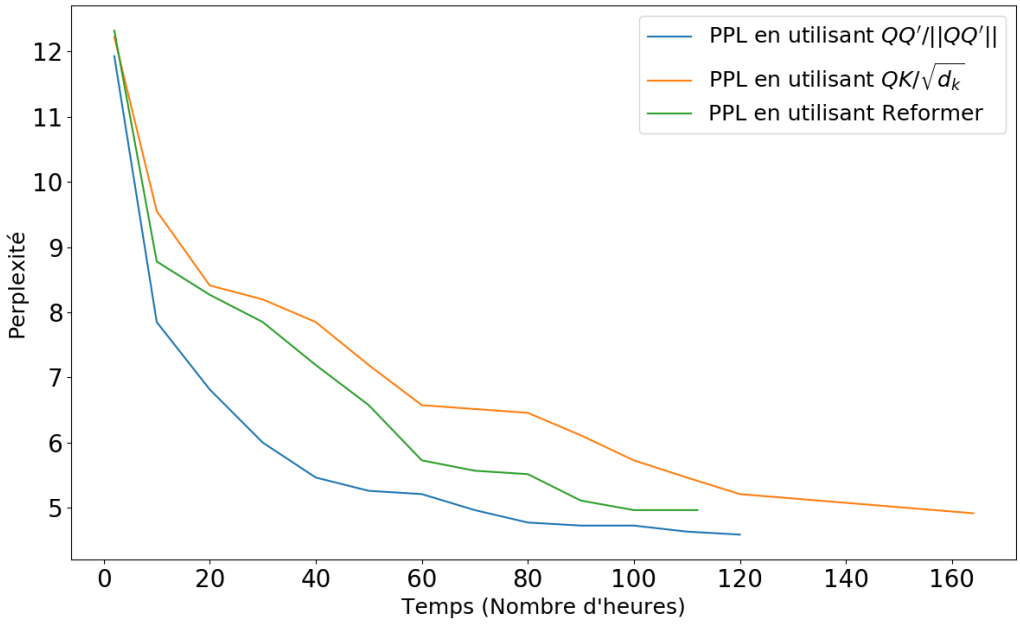


Figure 4.1: Illustration de l'évolution temporelle et de la meilleure convergence de la perplexité (*PPL*) pour la tâche de traduction automatique anglais-français, avec comparaison du modèle proposé à base du cosinus avec le Transformer et le Reformer

#Cauches Encoder/Décoder	Modèle	EN-GE			EN-FR		
		Entraînement (hrs)	**Étapes**(10^3)	**Itérations**	**Entraînement (hrs)**	**Étapes**(10^6)	**Itérations**
4	Transformer	57	600	16	157	3.9	13
	Reformer	**41**	**450**	**12**	**109**	**2.7**	**9**
	Notre Modèle	44	525	14	114	2.7	9
6	Transformer	58	675	18	164	4.8	16
	Reformer	**43**	**450**	**12**	**112**	**3**	**10**
	Notre Modèle	46	600	16	120	3.6	12

Tableau 4.3: Les statistiques d'entraînement pour les tâches de traduction automatique (WMT-2014) anglais-allemand et anglais-français

couches d'encodeur et de décodeur à 4 et 6. Cette table résume les résultats de la traduction, les quatre premières colonnes relatives à l'ensemble des tests et les quatre

#Couches Encodeur/Décodeur	Model	Ensemble Test				Ensemble Développement			
		PPL	**BLEU**	**WER**	**ACW**	**PPL**	**BLEU**	**WER**	**ACW**
		Traduction automatique anglais-allemand							
4	Transformer	5.87	25.64	0.67	0.67	5.34	26.57	0.66	0.68
	Reformer	5.89	25.51	0.68	0.66	5.34	26.34	0.67	0.68
	Notre Modèle	**5.73**	**26.13**	**0.66**	**0.69**	**5.27**	**26.84**	**0.66**	**0.71**
6	Transformer	5.81	27.28	0.67	0.68	5.17	28.03	0.66	0.69
	Reformer	5.84	27.26	0.69	0.66	5.19	27.68	0.67	0.68
	Notre Modèle	**5.69**	**28.19**	**0.65**	**0.72**	**5.04**	**28.52**	**0.65**	**0.72**
		Traduction automatique anglais-français							
4	Transformer	5.84	37.81	0.65	0.69	5.08	39.91	0.64	0.69
	Reformer	5.81	37.13	0.67	0.68	5.11	39.36	0.65	0.67
	Notre Modèle	**5.76**	**38.93**	**0.64**	**0.72**	**4.81**	**39.96**	**0.63**	**0.72**
6	Transformer	5.71	38.06	0.64	0.71	4.94	39.97	0.63	0.71
	Reformer	5.74	37.74	0.65	0.69	4.98	39.86	0.64	0.70
	Notre Modèle	**5.42**	**40.32**	**0.63**	**0.75**	**4.21**	**41.94**	**0.62**	**0.76**

Tableau 4.4: Résultats de la tâche de traduction automatique anglais-allemand et anglais-français en termes de perplexité, de score *BLEU*, du taux d'erreurs des mots (*WER*) et du taux des mots corrects (*ACW*) en utilisant des encodeurs et des décodeurs à 4 et à 6 couches

autres à l'ensemble des développements. Comme déjà mentionné, le même code et les mêmes données ont été utilisés afin de comparer le transformer avec notre modèle, à l'exception des parties de la représentation des données de l'attention. Pour le Reformer, la différence comprenait également l'utilisation du LSH et de ses optimisations de calcul (Kitaev *et al.*, 2020).

Le tableau 4.3 montre que le temps d'entraînement le plus court a été atteint par le Reformer, avec celui du modèle proposé relativement proche de celui du transformer. La Figure 4.1 compare la convergence des meilleures perplexités de la validation pour les

trois modèles pendant l'apprentissage. Elle montre une valeur de 4.2 après 120 heures d'apprentissage pour notre modèle contre près de 4.94 et 4.98 pour le transformer et le Reformer respectivement. De plus, la figure montre que le modèle proposé présente une décomposition plus douce et plus monotone.

Concernant les performances de la traduction, le tableau 4.4 montre que, pour la tâche de traduction automatique utilisant les deux paires de langues anglais-allemand et anglais-français, le modèle proposé a surpassé le transformer et le Reformer en termes des métriques utilisées. Le Reformer a donné les pires résultats. Ainsi, le score *BLEU* du modèle proposé pour les tâches de traduction automatique anglais-français et anglais-allemand ont montré une amélioration d'environ 2 points *BLEU* par rapport aux deux autres modèles. De plus, notre modèle à 4 couches a fourni une meilleur performance à celle du transformer ayant 6 couches après seulement 2 jours d'entraînement au lieu de 7. Ainsi, il ya eu un gain considérable en terme de temps pour notre modèle proposé.

Le taux d'erreurs de mots est le rapport entre le nombre d'éditions et la taille de la référence. Ces éditions sont appliquées aux phrases d'hypothèse afin de les rendre équivalentes à leurs références (Popović et Ney, 2007). On peut remarquer que le résultat du taux d'erreur de mots est en corrélation avec les résultats obtenus par le score *BLEU* et la perplexité. Nous notons qu'il existe une relation inverse entre le score *BLEU* et le taux d'erreur et entre le score *BLEU* et la perplexité. Plus le score *BLEU* de notre modèle augmente par rapport aux autres modèles, plus la perplexité et le taux d'erreur de notre modèle diminuent.

Dans la traduction automatique de l'anglais vers l'allemand, notre modèle avait le taux d'erreur le plus bas de 0,65, suivi de transformer de base de 0,67 et en dernier le Reformer avait 0,69. Même remarque pour la traduction automatique anglais-français, notre modèle a obtenu le taux d'erreur le plus bas de 0.63 suivi de transformer de base avec 0.64 et le Reformer se classe en dernier avec 0.65.

Pour la perplexité, la traduction automatique de l'anglais vers l'allemand de notre modèle avait une perplexité la plus faible de 5.69, suivi de transformer de base avec 5.81 et le Reformer en dernier avec 5.84. Même remarque pour la traduction automatique anglais-français, notre modèle a obtenu la perplexité la plus faible de 5.42, suivi de transformer de base avec 5.71 et le Reformer en dernier avec 5.74.

De plus, le score *BLEU* obtenu par notre modèle est le plus élevé dans la traduction automatique anglais-allemand qu'est égal à 28.19, suivi de transformer de base avec 27.28, et en dernier le reformer avec 27.26. Même remarque pour la traduction automatique anglais-français, le meilleur résultat a été obtenu par notre modèle de 40.32, suivi de transformer de base avec 38.06 et le Reformer en dernier avec 37.74.

4.3.4 Analyse des erreurs

En examinant la traduction automatique de l'ensemble newtest2014 des trois modèles, nous avons remarqué plusieurs types d'erreurs qui peuvent être résumées dans les exemples du tableau 4.5. Les mots en gras dans le tableau représentent les mots générés par les modèles et manquants de la référence.

Le premier exemple nous montre un échec des trois modèles pour la traduction des mots hors vocabulaire manquants dans l'ensemble d'entraînement, tels que le mot « 90M$ », ou des nombres tels que « 10 000 ». Ces mots hors vocabulaire ont un impact négatif sur le résultat du score *BLEU*. Une autre remarque est la possibilité de traduire un mot d'une phrase source avec un synonyme qui n'appartient pas à la phrase de référence. À titre d'exemple, le mot «voitures» et son synonyme «véhicules» ; sont considérés tous deux comme étant des traductions possibles du mot « *Cars* ». La traduction proposée par le Reformer ainsi que de notre modèle est le mot « véhicules ». Ces erreurs de traduction font baisser le score *BLEU*, le taux d'erreur et même la perplexité.

Dans le deuxième exemple, nous pouvons remarquer que notre modèle a réussi à tra-

Exemple 1	**Langue source**	The U.S. Senate approved a $90-million pilot project last year that would have involved about 10,000 cars.
eN-FR	**Traduction**	
	Transformer	le sénat américain a approuvé l'année dernière un projet pilote de **<UNK>** qui aurait **impliqué** environ **<UNK>** véhicules .
	Reformer	le sénat américain a approuvé l'année dernière un projet pilote de **<UNK>** qui **impliquerait** environ **<UNK>** voitures .
	Notre Modele	le sénat américain a approuvé l'année dernière un projet pilote de **<UNK>** de dollars qui aurait **impliqué** environ **<UNK>** véhicules .
	Référence	Le Sénat américain a approuvé un projet pilote de 90 M$ l'année dernière qui aurait porté sur environ 10 000 voitures.
Exemple 2	**Langue source**	the Thin White Duke was also planning to re-release the album on November 04.
EN-FR	**Traduction**	
	Transformer	le **duc blanc mince prévoit** de **éditer** le 4 novembre cet album .
	Reformer	le **duc blanc mince prévoit** de **éditer** le 4 novembre **cette** album.
	Notre Modele	le **duc blanc mince prévoit** de rééditer cet album le 4 novembre .
	Référence	Le Thin White Duke a ainsi prévu de rééditer cet album, le 4 novembre.
Exemple 3	**Langue source**	"It varies from one quarter to the next in the face of very lively competition."
EN-FR	**Traduction**	
	Transformer	« **elle** varie d'un trimestre à l'autre **face** à une concurrence très vive.
	Reformer	« **elle** varie d'un trimestre à l'autre **face** à une concurrence très vive.
	Notre Modele	« **il** varie d'un trimestre à l'autre **face** à une concurrence très vive. »
	Référence	Ça varie d'un trimestre à l'autre avec une concurrence très vive.
Exemple 4	**Langue source**	Kirchen-Hausen will celebrate the 1,250th anniversary with a celebratory weekend from 18 to 20
EN-DE	**Traduction**	
	Transformer	**<UNK>** feiert **das <UNK> jubiläum mit** einem festwochenende vom 18 bis 20(UNK célèbre l'anniversaire de UNK avec un week-end de festival du 18 au 20)
	Reformer	**<UNK>** feiert **das <UNK> jubiläum mit** einem festwochenende vom 18 bis 20(*UNK célèbre le UNK anniversaire avec un week-end du festival du 18 au 20*)
	Notre Modele	**<UNK>** wird im juli an einem festwochenende vom 18 bis 20 juli die **<UNK>**. (*UNK sera l'UNK lors d'un week-end de festival du 18 au 20 juillet.*)
	Référence	Kirchen-Hausen wird im Juli an einem Festwochenende vom 18. bis 20. Juli die 1250.
Exemple 5	**Langue source**	They can expect prison terms of up to ten years.
EN-DE	**Traduction**	
	Transformer	**ihnen** müssen mit haftstrafen von bis zu **10** jahren rechnen. (*ils font face à des peines de prison allant jusqu'à 10 ans.*)
	Reformer	**ihnen** müssen mit haftstrafen von bis zu **10** jahren.(*Ils font face à des peines d'emprisonnement allant jusqu'à 10 ans.*)
	Notre Modele	**ihnen drohen** haftstrafen **von** bis zu **10** jahren rechnen.(*ils font face à des peines d'emprisonnement allant jusqu'à 10 ans.*)
	Référence	Sie müssen mit Haftstrafen bis zu zehn Jahren rechnen.

Tableau 4.5: Exemples de traduction générée par les modèles sur les corpus des tests anglais-français et anglais-allemande.

duire le mot « *re-release* » dans son contexte avec le mot « rééditer ». Cependant, les autres modèles ont traduit le mot « *re-release* » avec le mot « éditer » qui est différent du mot présenté dans la référence qui est «rééditer». Dans cet exemple, l'expression « *Thin White Duke* » désigne un nom d'un groupe musical. Pour cette raison, l'expression n'a pas été traduite dans la référence. En revanche, tous les modèles ont tenté de traduire cette expression, et peuvent donc affecter le résultat du score *BLEU*, le taux d'erreur des mots et la perplexité.

Le troisième exemple montre un autre problème existant dans la référence proposée dans le corpus *newtest2014* lui-même, où cette dernière n'inclut pas les guillemets. Bien que la phrase source ait des accolades, la phrase de la référence proposée ne présente aucun guillemet. Cela affecte également le score *BLEU*, car les guillemets sont considérés comme des mots pour les modèles et ils sont inclus dans le vocabulaire. Dans cet exemple, notre modèle a réussi quand même à fermer le guillemet droit contrairement aux autres modèles.

Les exemples 4 et 5 sont des extraits de la traduction anglais-allemand de la collection *newtest2014*. Pour plus de lisibilité, les phrases générées pour les modèles ont été traduites en français à l'aide de Google Traduction et écrites en italique. L'exemple 4 nous montre que les traductions de mots hors vocabulaire (en anglais, out-of-vocabulary) ont été remplacées par *UNK*. De plus, on peut voir que le Transformer et le Reformer ont suggéré des mots qui n'existent pas dans la référence comme « *Das Jubiläum mit* » qui signifie « l'anniversaire avec ». En effet, ce mot n'a aucune relation avec le sens de la phrase traduite. Cependant, notre modèle a traduit la phrase source en proposant tous les mots existants dans la phrase cible sauf « *Kirchen-Hausen* » qui représente une entité nommée en tant que mot composé, et « 1250 » qui représente un nombre, donc une autre entité nommée.

Les entités nommées, dont les nombres, ne représentent pas des ensembles fermés.

C'est pourquoi nous ne pouvons pas les inclure toutes dans l'ensemble du vocabulaire. L'ensemble du vocabulaire ne comprend que les entités nommées et les nombres existants dans les corpus d'apprentissage avec une fréquence de plus que 3 répétitions dans le corpus, comme on peut remarquer avec les nombres « 18 » et « 20 » de l'exemple 4 et « 10 » de l'exemple 5.

Dans l'exemple 5, on peut voir que le mot « *ten* » a été traduit dans la référence par le mot « *zehn* » qui signifie « dix ». Cependant, tous les modèles ont traduit le mot en nombres de « 10 ». Bien que cette traduction soit toujours correcte, le mot « 10 » n'existe pas dans la référence, ce qui peut réduire le score *BLEU* et faire augmenter le taux d'erreurs des mots (*WER*) et la perplexité.

4.3.5 Discussion

En utilisant un modèle à 6 couches avec une taille de lot de 1024, le Reformer tel que présenté par Kitaev *et al.* (2020) a rapporté un score *BLEU* sur l'ensemble newstest2014 de la tâche WMT anglais-allemand [27] qui est inférieur à celui obtenu par notre modèle avec une taille de lot de 120, tous les autres paramètres étant identiques (27,6 *BLEU* contre 28,19 *BLEU*). Cependant, Kitaev *et al.* (2020) n'ont pas évalué le corpus WMT anglais-français. C'est pourquoi nous avons évalué nous-mêmes ce corpus et nous avons obtenu un score *BLEU* de 37.74 pour le reformer contre 40.32 obtenu par notre modèle.

Nous avons utilisé le rééchantillonnage bootstrap comme décrit dans Koehn (2004) pour évaluer la signification statistique et étudier la stabilité de cette différence en *BLEU* et la *Perplexité*. Ce rééchantillonnage aléatoire avec remplacement a été mis en œuvre par la création de 100 ensembles de tests ayant la moitié de la taille de ceux de newtest 2014 (d'où 1501 phrases dans chacun). Ensuite, nous avons calculé les scores *BLEU* et ceux de la *perplexité* pour chaque ensemble de tests par notre approche, soit

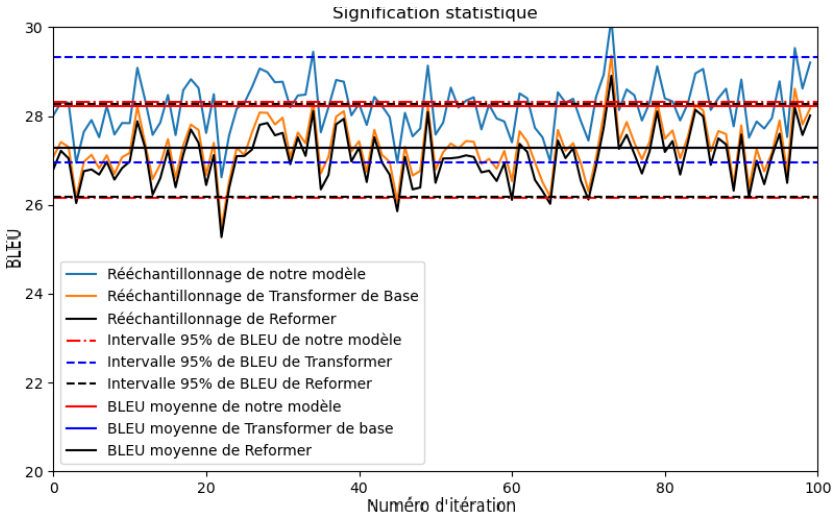


Figure 4.2: Résultats du rééchantillonnage Bootstrap pour 100 ensembles de tests dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-allemand. Les moyennes et intervalles de confiance des Scores *BLEU* calculés entre notre modèle, le Reformer et le Transformer

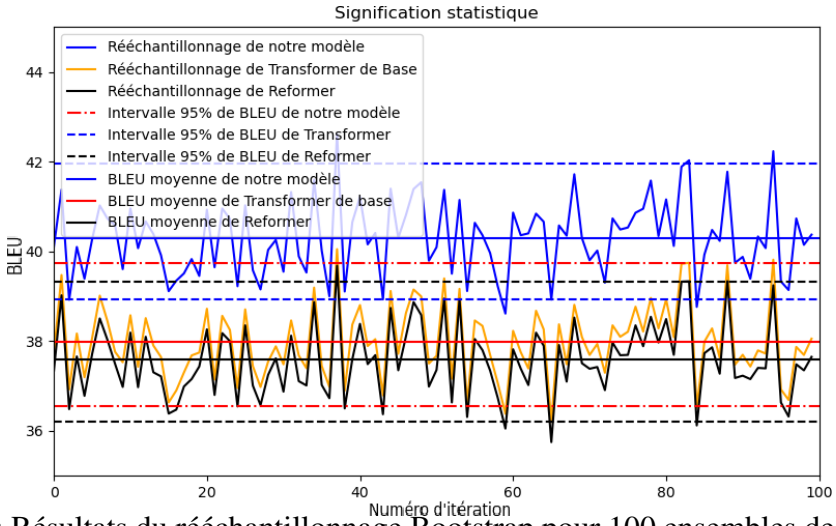


Figure 4.3: Résultats du rééchantillonnage Bootstrap pour 100 ensembles de tests dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-français. Les moyennes et intervalles de confiance des Scores *BLEU* calculés entre notre modèle, le Reformer et le Transformer.

le Transformer de base et le Reformer.

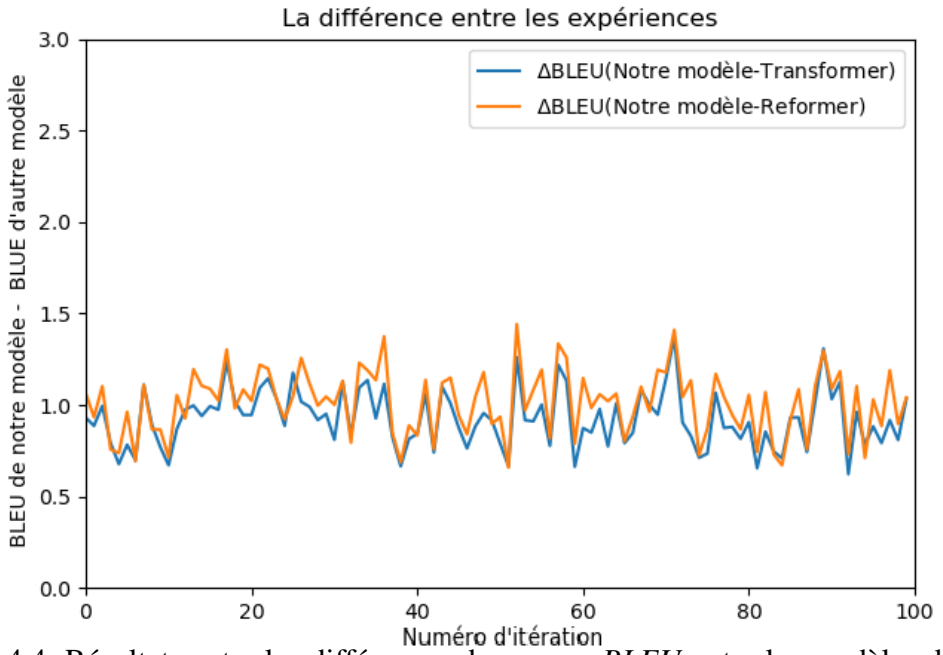


Figure 4.4: Résultats entre les différences des scores *BLEU* entre les modèles dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-allemand.

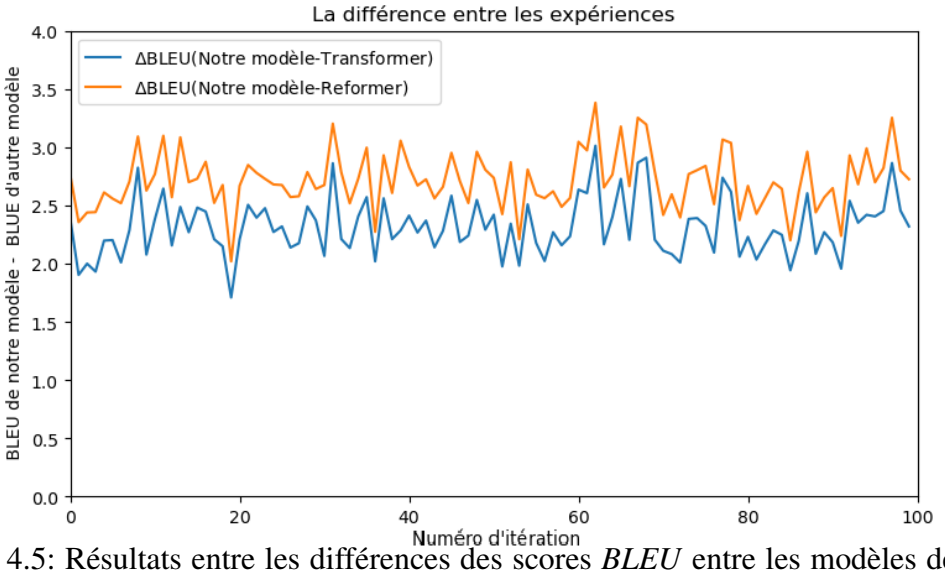


Figure 4.5: Résultats entre les différences des scores *BLEU* entre les modèles dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-français.

Les figures 4.2 et 4.3 fournissent des vues détaillées de notre modèle, du Reformer et du Transformer en utilisant 100 ensembles de tests dérivés de chaque ensemble newstest2014 de la tâche WMT-2014 (anglais-allemand, anglais-français), avec des

graphes des scores *BLEU* obtenus, leurs intervalles de confiance à 95 % et leurs valeurs moyennes.

Les résultats des tests appliqués sur la tâche WMT-2014 anglais-allemand montrent des intervalles de confiance à 95 % de [27,0, 29,3] pour notre modèle, [26,2, 28,2] pour le Reformer et [26,2, 28,3] pour le Transformer, avec des valeurs moyennes de 28,20, 27,20 et 27,20 respectivement. Ces valeurs sont très proches de celles rapportées dans le tableau 4.4 pour l'ensemble de test newtest2014 : 28,19 et 27,26 respectivement.

Les résultats des tests appliqués sur la tâche WMT-2014 anglais-français montrent des intervalles de confiance à 95 % de [38.9, 42.0] pour notre modèle, [36.2, 39.3] pour le Reformer et [36.6, 39.7] pour le Transformer, avec des valeurs moyennes de 40,20, 37,60 et 38,0 respectivement. Ces valeurs sont très proches de celles rapportées dans le tableau 4.4 pour l'ensemble de test newtest2014 anglais-français : 28,19 et 27,26 respectivement.

Les figures 4.4 et 4.5 montrent les graphes de la différence entre les scores *BLEU* obtenus par notre modèle comparé à ceux du Transformer de base et du Reformer. Elles montrent aussi une meilleure performance de notre modèle pour les 100 échantillons de tests.

Les figures 4.6 et 4.7 montrent les graphes de la différence entre les scores de la *perplexité* obtenus par notre modèle comparé à ceux du Transformer et du Reformer en utilisant 100 ensembles de tests dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-allemand. Elles montrent aussi une meilleure performance de notre modèle pour les 100 échantillons de tests.

Les figures 4.6 et 4.7 fournissent des vues plus détaillées de notre modèle, du Reformer et du Transformer, avec des graphes des scores de la *Perplexité* obtenus, leurs intervalles de confiance à 95 % et leurs valeurs moyennes.

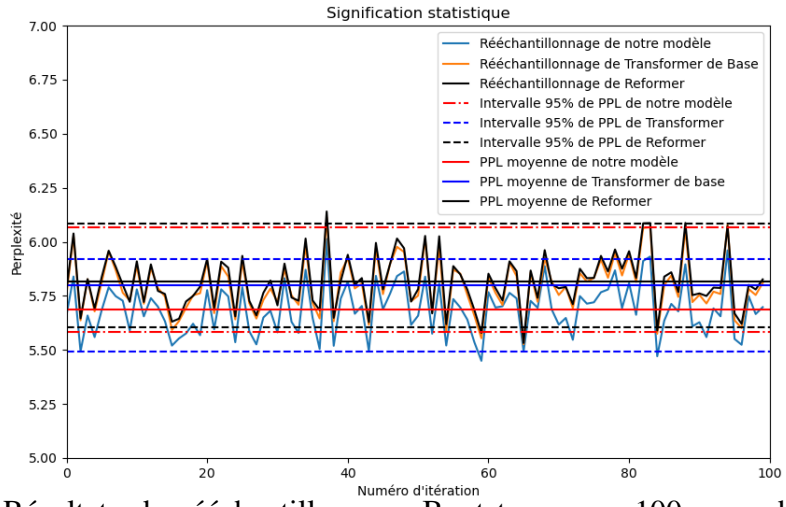


Figure 4.6: Résultats du rééchantillonnage Bootstrap pour 100 ensembles de tests dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-allemand. Les moyennes et intervalles de confiance des Scores de la *Perpelexité* calculés entre notre modèle, le Reformer et le Transformer

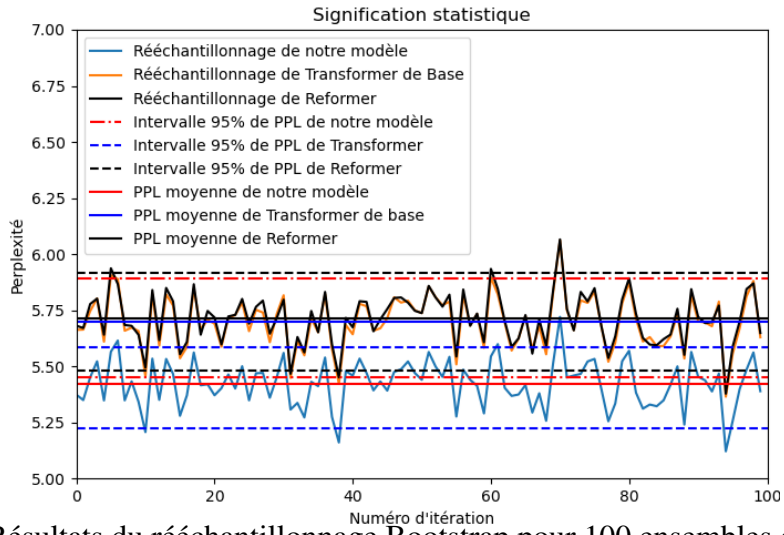


Figure 4.7: Résultats du rééchantillonnage Bootstrap pour 100 ensembles de tests dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-français. Les moyennes et intervalles de confiance des Scores de la *Perpelexité* calculés entre notre modèle, le Reformer et le Transformer

Dans la figure 4.6, les résultats de l'ensemble de tests anglais-allemand montrent des intervalles de confiance à 95 % de [5.5, 5.9s] pour notre modèle, [5.6, 6.1] pour le Re-

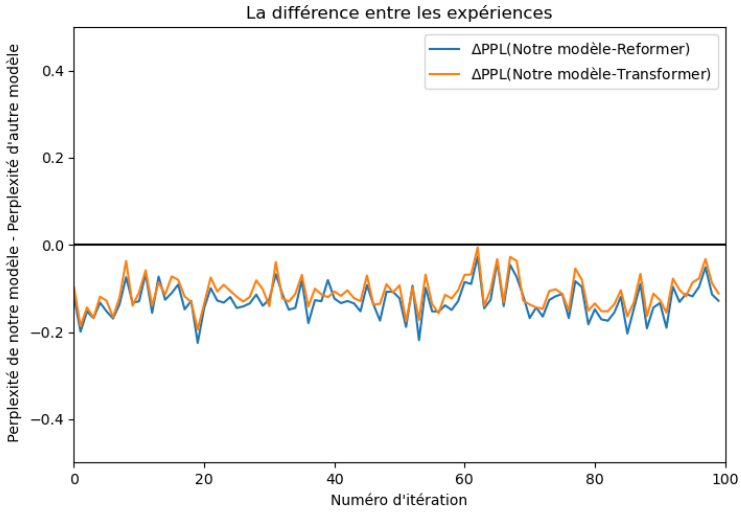


Figure 4.8: Résultats entre les différences des scores *Perplexité* entre les modèles dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-allemand.

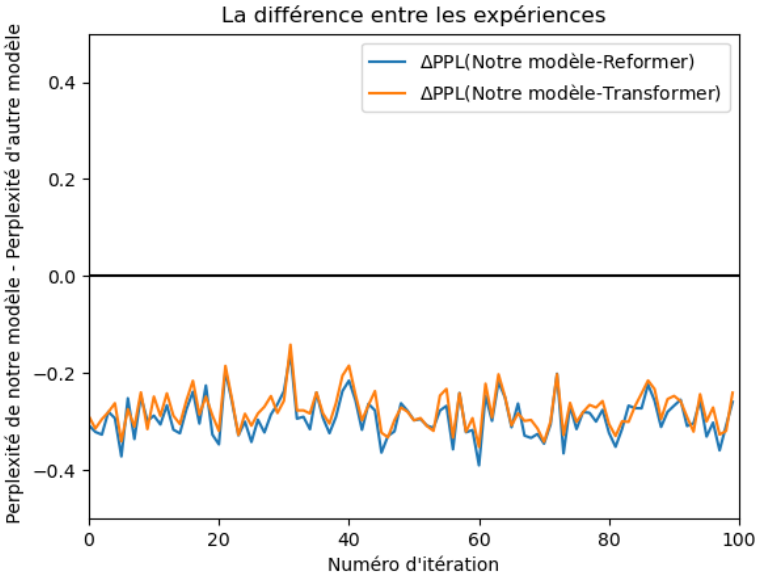


Figure 4.9: Résultats entre les différences des scores *Perplexité* entre les modèles dérivés de l'ensemble newstest2014 de la tâche WMT-2014 anglais-français.

former et [5.5, 6.0] pour le Transformer, avec des valeurs moyennes de 5,7, 5,9 et 5,8 respectivement. Ces valeurs sont très proches de celles rapportées dans le tableau 4.4 pour l'ensemble de tests newtest2014 anglais-allemand : 5.69, 5.84 et 5.81 respective-

ment.

Par ailleurs, la figure 4.7 montre des intervalles de confiance à 95 % de [5.2, 5.6] pour notre modèle, [5.5, 5.9] pour le Reformer et [5.5, 5.8] pour le Transformer, avec des valeurs moyennes de 5.4, 5.7 et 5.7 respectivement. Ces valeurs sont très proches de celles rapportées dans le tableau 4.4 pour l'ensemble de tests newtest2014 anglais-français : 5.42, 5.74 et 5.71 respectivement.

Les résultats des figures 4.8 et 4.9 montrent que pour tous les résultats des échantillonnages, la perplexité de notre modèle est inférieure à celle du Transformer et du Reformer pour les deux ensembles de tests.

Les meilleures performances de traduction de notre modèle expliquent que sa génération de valeurs d'attention est plus importante que celle du Transformer. La figure

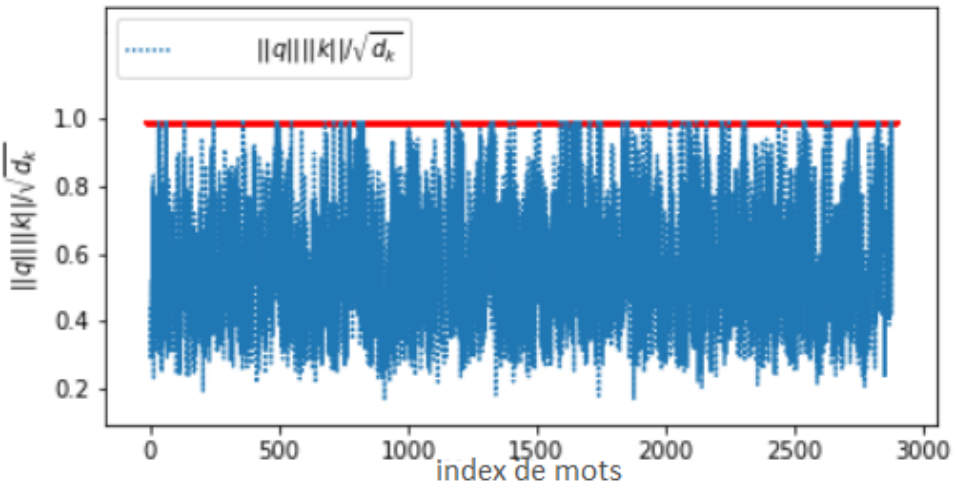


Figure 4.10: Exemple de distribution du $\|q\|\|k\|/\sqrt{d_k}$ ratio lors du traitement d'un lot du jeu de développement anglais-français WMT-2014

4.10 montre un exemple de distribution de $\|q\|\|k\|/\sqrt{d_k}$ ratio avec $d_k = 64$. Cela a été généré à partir de l'attention croisée de la dernière couche. La courbe révèle que le $\|q\|\|k\|$ produit est lié par $\sqrt{d_k}$, puisque le rapport ne dépasse jamais 1. Par conséquent,

nous avons toujours $\frac{qk^T}{\sqrt{d_k}} \leq cos(q\acute{q}^T) \leq 1$.

Kitaev *et al.* (2020) rapportent que l'entraînement du Reformer a été parallélisée sur 8-cores GPU / TPU v3, avec une capacité de mémoire totale de 128 Go, bien supérieure aux 11 Go de la carte GPU Nvidia GTX1080 Ti que nous avons utilisée. Nous avons étudié l'impact potentiel de cette différence de mémoire en exécutant le Reformer sur notre plate-forme et en l'appliquant à la fois à la tâche WMT2014 anglais-allemand et à la tâche WMT2014 anglais-français plus exigeante. Nos résultats ont encore donné l'avantage au Reformer pour le temps d'entraînement. Cependant, le gain par rapport à notre modèle n'était que de 6,8 % et 4,3 % respectivement, comme le montrent les résultats obtenus du tableau 4.3. C'est moins que le gain théorique attendu $O(l^2)$ à $O(l\ log(l))$ qui est rapporté dans Kitaev *et al.* (2020). Cela peut être dû à l'exigence du Reformer : La longueur de la séquence est divisible par le double de la taille du bucket [19]. Ceci force le modèle à remplir fréquemment les séquences traitées pour répondre à cette exigence, avec un impact négatif sur le temps d'apprentissage.

Le Reformer a également obtenu des performances d'apprentissage plus rapides grâce à l'utilisation du traitement de blocs, au hachage sensible à la localité et aux couches résiduelles réversibles pendant l'apprentissage. La technique des couches résiduelles réversibles peut également être appliquée à notre modèle et nous prévoyons des gains en termes de vitesse de l'entraînement une fois ces couches sont mises en œuvre. Ce qui peut faire gagner à notre modèle un peu de vitesse d'apprentissage. En revanche, l'utilisation de l'algorithme de hachage n'est pas applicable à notre architecture.

De plus, ces modifications peuvent augmenter le nombre de paramètres rendant notre modèle plus difficile à généraliser pour les domaines de dialogue multi-tours, comme nous le verrons plus loin.

---

19. https ://github.com/lucidrains/reformer-pytorch#additional-helpers

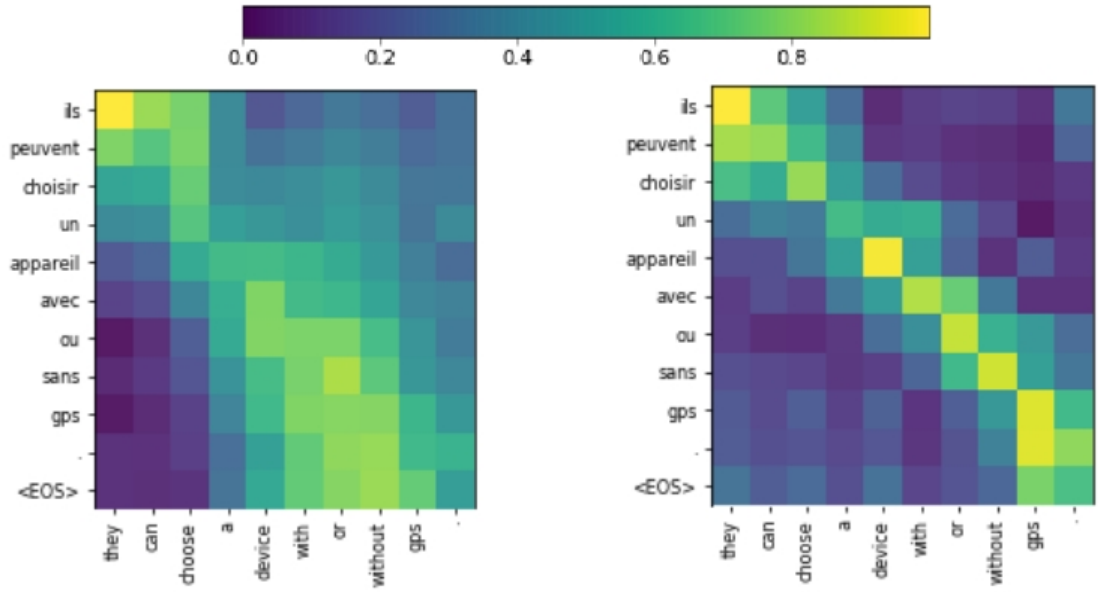


Figure 4.11: Illustration pour la tâche anglais-français de l'attention moyenne de la tête à la limite encodeur-décodeur dans le Transformer (à gauche) et notre modèle (à droite).

Enfin, la figure 4.11 montre des graphiques, pour la tâche anglais-français, de la moyenne des dernières têtes de l'attention entre encodeur-décodeur du transformer et de notre modèle (sixième couche), pris sur toutes les têtes. Le tracé de gauche correspond à l'attention du produit scalaire à l'échelle du transformer et celui de droite à notre attention basée sur la similitude cosinus. La figure montre que notre attention a tendance à être plus concentrée, avec un maximum plus clair pour la plupart des positions de pairs source-cible, tandis que l'attention du transformer est plus diffuse. Par exemple, dans le transformer de base, la figure montre le mot «*choose*» en relation avec plusieurs mots comme «ils peuvent choisir un» en même temps et avec la même intensité de couleur, tandis que le mot «*choose*» dans l'attention proposée de notre modèle a une relation directe avec le mot «choisir» avec une intensité de couleur plus forte. Cette relation directe reflète la réalité où la traduction du verbe «*choose*» en français devrait être un autre verbe qui est le mot «choisir».

4.4 Évaluation du modèle de dialogue

Nous avons utilisé trois ensembles de données de tailles et de formats différents (réseaux sociaux, sous titrage de films et forums de discussions), ainsi que plusieurs mesures de performance afin d'évaluer les performances de conversation ouverte de Cascade et le transformer MDA. Nous avons également mené ces expériences avec six modèles de pointe à des fins de comparaison :

1. Le Transformer comme modèle de base (Vaswani *et al.*, 2017).
2. L'encodeur-décodeur récurrent hiérarchique (HRED) (Serban *et al.*, 2016b) .
3. L'encodeur-décodeur récurrent hiérarchique variationnelle (VHRED) (Serban *et al.*, 2017c)
4. Les RNRs de conversation hiérarchique variationnelle (VHCR) (Zhao *et al.*, 2017)
5. Le réseau hiérarchique d'attention récurrente (HRAN) (Xing *et al.*, 2018)
6. Le contexte pertinent avec le modèle d'auto-attention (ReCoSa) (Zhang *et al.*, 2019a).

4.4.1 Prétraitement des données

Pour tous les ensembles de données, nous avons limité les énoncés à 30 mots et tronqué ceux dépassant cette limite. Comme plus de 96% des énoncés dans le corpus le plus grand (Ubuntu) étaient de cette longueur ou moins, le réglage des capacités d'entrée de l'encodeur et des décodeurs à 30 mots permettait une meilleure efficacité de calcul.

Chaque énoncé a été analysé pour la tokenisation de mot et la reconnaissance des entités nommées à l'aide de l'outil spacy [20], les noms et les nombres étant respectivement remplacés par les jetons <Person>, <Organisation>, <Location>, <Monnaie>, <Date> et <Nombre> comme cela est fait dans Ritter *et al.* (2010) . Ce remplacement vise à

20. spacy.io

réduire les problèmes de rareté des mots dans le corpus (Serban *et al.*, 2016a). De plus, tous les mots ont été convertis en lettres minuscules pour la réduction des données, et ceux étant apparus moins de trois fois ont été remplacés par le jeton générique <UNK> afin de donner au modèle le pouvoir de traiter les mots hors du vocabulaire durant l'inférence. Cela a abouti à des tailles de vocabulaire de 169k pour Ubuntu, 150k pour Cornell Movie et 78k pour DailyDialog.

Les conversations du corpus sont divisées en ensembles d'entraînement, de validation et de test. Le corpus Ubuntu est divisé par l'outil [21]. Les autres corpus sont déjà divisés par leurs auteurs respectifs. Comme indiqué dans le tableau 4.6. Comme déjà mentionné,

Corpus	**Entraînement**	**Validation**	**Test**
DailyDialog	11118	1000	1000
Cornell Movie	79879	1600	1600
Ubuntu	892 000	19 000	19 000

Tableau 4.6: La taille des ensembles d'entraînement, de validation et de test, après la division de chaque corpus.

Cascade utilise des séquences de dialogue à 4 tours au maximum (E_1, E_2, E_3, E_4), où E_1 et E_3 sont des requêtes de l'utilisateur et E_2 avec E_4 des réponses système. De plus,les trois premiers tours servent à prédire le quatrième et les deux premiers peuvent contenir juste un jeton vide (<Pad>). De la même manière que Serban *et al.* (2016b), nous ajoutons un jeton de fin d'énoncé à tous les énoncés pouvant passer dans les décodeurs.

Les autres modèles avec le Transformer MDA ont été entraînés sur des dialogues allons de 1 jusqu'à 10 tours, et avec un nombre total de couches égale à 6 pour l'encodeur et

21. https ://github.com/rkadlec/ubuntu-ranking-dataset-creator

le décodeur comme le Transformer de base (Vaswani *et al.*, 2017).

Cascade utilise 4 tours pour le test dans un tour à prédire comme réponse, et l'historique allons de 1 jusqu'à 3 tours. Cependant, les autres modèles utilisent une réponse et de 1 à 10 tours dans l'historique [22].

Pour le transformer de base, nous avons concaténé tous les énoncés de l'historique de dialogue sous la forme d'une longue séquence représentant une requête. Ensuite, cette requête accompagnée de la réponse concernée a été traitée comme une paire question-réponse. Par ce moyen, nous avons transformé le problème de la génération de réponses multi-tours en un problème de génération de réponses d'un seul tour afin d'utiliser le transformer comme base de référence.

Pour une comparaison équitable, la taille de plongement qui était de 256 pour ReCoSa et Transformer égale au vecteur de sortie de l'auto-attention. Le nombre d'unités cachées était de 300 pour tous les autres modèles [23]. La taille 300 de plongement est fréquemment utilisée pour les modèles de l'état de l'art (Fan *et al.*, 2019; Serban *et al.*, 2017b; Huang *et al.*, 2018). La valeur différente de ReCoSa était due à son utilisation d'un multiple de 8, son nombre de têtes d'attention. La taille du lot a été fixée à 80 pour tous les ensembles de données, sauf pour Ubuntu où elle a été réduite à 32 en raison des limites de consommation de mémoire imposées par la taille de son vocabulaire. Les autres paramètres ont également été réglés sur les mêmes valeurs (Park *et al.*, 2018; Xing *et al.*, 2018). La taille utilisée dans l'algorithme de la recherche en faisceau (*Beam search* (Meister *et al.*, 2020)) pour tous les modèles est égale à 4.

Tous les modèles ont été encodés en utilisant PyTorch et fonctionnent sur une carte

---

22. Nous n'avons pas entraîné Cascade sur plus de tours en raison de nos ressources informatiques limitées

23. Nous utilisons les plongements de Google News : https ://code.google.com/archive/p/word2vec/.

GPU Nvidia GTX1080 TI, dotée de 3584 cours de traitement FP32 et de 11 Go de mémoire GDDR5, pour une vitesse de traitement allant jusqu'à 11,4 TFLOP et jusqu'à 484 Go de bande passante de mémoire.

4.4.2 Modèles de référence et mesures de performance

Nous avons utilisé à la fois des évaluations automatiques et humaines pour les comparaisons, car aucune métrique d'évaluation automatique standard, qui est en corrélation avec le jugement humain pour les systèmes de génération de réponses pour les systèmes de dialogue à domaine ouvert, n'existe encore et ce malgré les mesures prometteuses dans les travaux de l'état de l'art récent Pang *et al.* (2020); Mehri et Eskénazi (2020); Sato *et al.* (2020).

Les évaluations automatiques incluaient la métrique *BLEU* (Papineni *et al.*, 2002), la perplexité (Serban *et al.*, 2017c) et les trois mesures de similarité basées sur les plongements des mots proposés (Liu *et al.*, 2016), à savoir la moyenne des plongements (Mitchell et Lapata, 2008), les vecteurs d'inclusion extrême (Forgues *et al.*, 2014) et la correspondance gourmande (Mitchell et Lapata, 2008; Rus et Lintean, 2012a).

Ainsi, les métriques basées sur les vecteurs de plongement ont évalué la similitude entre les mots de la réponse du modèle proposé et la vérité terrain (Xing *et al.*, 2018; Serban *et al.*, 2017c; Rus et Lintean, 2012b). La métrique moyenne des plongements projette chaque réponse sur un vecteur en prenant la moyenne de tous les plongements afin de calculer la similitude cosinus entre le vecteur de réponse du modèle et le vecteur de vérité terrain. La métrique des vecteurs extrêmes est similaire, sauf si elle prend l'extremum de chaque dimension au lieu de la moyenne. La métrique de la correspondance gourmande trouve d'abord le meilleur alignement de mots non exclusifs entre la réponse du modèle et la vérité terrain, puis calcule la moyenne sur la similarité cosinus

entre les mots alignés [24].

La métrique de la perplexité mesurait la capacité du modèle à prendre en compte la structure syntaxique du dialogue et celle de chaque énoncé généré (par exemple, les signes de ponctuation et la distribution des mots) (Xing *et al.*, 2018; Liu *et al.*, 2016). Sur cette base, le modèle avec la plus faible perplexité est considéré comme le système de génération de réponse le plus fluide. Dans nos expériences, la perplexité lors de la validation a été utilisée pour déterminer quand arrêter l'entraînement. Si la perplexité cesse de diminuer, nous pensons que l'algorithme a atteint la convergence et a terminé l'entraînement. Nous avons testé la capacité de génération de différents modèles avec la mesure de perplexité sur les données de test.

L'évaluation humaine a consisté en une enquête menée par cinq personnes à qui on a demandé de classer les réponses des différents modèles en fonction de la solidité sémantique et de la pertinence du sujet. Pour chaque modèle et corpus, 100 échantillons de réponses de dialogue ont été choisis au hasard et les évaluateurs ont été invités à classer les réponses générées sans connaître les modèles ou les corpus d'entraînement. Ensuite, les moyennes des victoires, des défaites et des égalités de chaque corpus et modèle, pour les cinq évaluateurs, ont été calculées pour chaque corpus et chaque modèle.

4.5 Évaluation du modèle Cascade

4.5.1 Évaluation automatique

Le tableau 4.7 compare les performances du modèle à celles des modèles de l'état de l'art sur les trois ensembles de données d'évaluation. Notre modèle proposé et nommé

24. Toutes les comparaisons étaient basées sur les plongements de Word2Vec entrainés par Google News Corpus : https ://code.google.com/archive/p/word2vec/.

Corpus	**DailyDialog**				**Ubuntu**				**Cornell Movie**			
Modèle	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL
Transformer	0.514	0.372	0.311	83.41	0.438	0.387	0.291	103.08	0.407	0.387	0.313	98.38
HRED	0.636	0.459	**0.391**	85.08	**0.638**	0.456	0.378	121.08	0.569	0.415	0.356	115.01
VHRED	0.620	0.432	0.363	96.73	0.549	0.400	0.308	192.93	0.557	0.402	0.352	186.79
VHCR	0.624	0.439	0.352	83.98	0.565	0.416	0.328	112.18	0.548	0.393	0.342	104.90
HRAN	0.611	0.419	0.314	83.48	0.568	0.429	0.364	101.44	0.512	0.387	0.332	97.41
ReCoSa	0.626	0.421	0.321	84.98	0.573	0.411	0.351	**97.16**	0.537	0.396	0.341	96.82
Cascade	**0.689**	**0.583**	**0.391**	**81.40**	0.584	**0.463**	**0.388**	102.03	**0.64**	**0.536**	**0.448**	**96.57**

Tableau 4.7: Les performances de prédiction de Cascade par rapport aux autres modèles pour trois corpus, en utilisant les métriques basées sur les plongements des mots et la perplexité. La similitude cosinus entre les réponses générées et réelles est donnée dans chaque cas.

Cascade [25] obtient les meilleurs résultats pour presque toutes les métriques. La seule exception notable est la métrique moyenne du corpus Ubuntu où HRED a fourni un meilleur résultat. Ces métriques nous donnent le rapprochement des vecteurs des réponses générées par rapport au vecteur de la réponse réelle. La métrique moyenne, par exemple, nous informe que la moyenne de chaque vecteur de la réponse générée est choisie avec des mots sémantiquement proches de la moyenne du vecteur de la réponse réelle.

Le tableau 4.7 rapporte aussi les résultats de perplexité nous informant sur la composition syntaxique. Il affiche systématiquement des valeurs inférieures pour Cascade par rapport aux autres modèles, à l'exception de l'ensemble de données Ubuntu où ReCoSa a fourni une valeur inférieure pour le corpus Ubuntu (Le corpus de grande taille). Ainsi, l'architecture simple encodeur-décodeur de Cascade, sans modules de traitement supplémentaires tels que des auto-encodeurs variationnels ou des encodeurs multiniveaux, semble être très efficace. Cependant, les valeurs de perplexité proche ou meilleure avec

25. https ://github.com/belainine/Cascade

HRAN et ReCoSa montrent que l'utilisation de l'attention dans une architecture bi-directionnelle ou de l'auto-attention dans l'encodeur et le décodeur, peut également conduire à des réponses fluides.

Corpus	**DailyDialog**				**Ubuntu**				**Cornell Movie**			
Modèle	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL
Seq2Seq(3 énoncés)	0.641	0.439	0.352	85.19	0.425	0.371	0.263	114.26	0.401	0.373	0.301	107.81
Cascade(1 dec)	0.611	0.419	0.314	87.84	0.568	0.429	0.364	116.43	0.392	0.367	0.312	109.41
Cascade(2 dec)	0.643	0.441	0.363	84.28	0.553	0.409	0.317	108.23	0.537	0.372	0.337	101.46
Cascade(3 dec sans contexte précédent)	0.613	0.408	0.321	88.36	0.559	0.414	0.367	117.89	0.381	0.358	0.307	109.73
Cascade(3 dec)	**0.689**	**0.583**	**0.391**	**81.40**	**0.584**	**0.463**	**0.388**	**102.03**	**0.640**	**0.536**	**0.448**	**96.57**

Tableau 4.8: Les performances de prédiction de Cascade utilisant 1 décodeur, 2 décodeurs et 3 décodeurs avec et sans historique de contexte par rapport au modèle Seq2Seq avec un encodeur utilisent 3 énoncés concaténés comme requête pour les trois corpus, en utilisant la perplexité et les métriques basées sur le plongement. La similarité en cosinus entre les réponses générées et réelles est donnée dans chaque cas.

Afin de prouver l'utilité des différents changements architecturaux, nous avons effectué plusieurs études d'ablation. Le tableau 4.8 compare les performances de Cascade dans le cas d'utilisation d'un décodeur, de deux décodeurs et de trois décodeurs, respectivement. Dans l'expérience réalisé avec un seul décodeur, le modèle apprend à générer un énoncé en utilisant une probabilité $P(E_2|E_1)$ où E_1 est l'historique de E_2 dans le corpus. Le modèle Cascade avec un seul decodeur est similaire au modèle seq2seq avec une attention du produit scalaire proposé par Luong *et al.* (2015a).

Dans l'expérience avec deux décodeurs, le modèle apprend à générer un énoncé E_3 en utilisant une probabilité $P(E_3|E_2,E_1)$ où E_1,E_2 représentent l'historique de E_3.

Enfin, dans la dernière expérience avec trois décodeurs, le modèle d'apprentissage a généré l'énoncé E_4 en utilisant une probabilité $P(E_4|E_3,E_2,E_1)$ où E_1,E_2,E_3 représentent l'historique de E_4.

Les résultats montrent que les performances de Cascade s'améliorent au fur et à mesure que le nombre de décodeurs augmente. La figure 4.12 montre la dépendance entre le nombre de décodeurs et le résultat de perplexité.

En outre, une comparaison a été faite avec un modèle Seq2Seq qui utilise l'attention des produits scalaires sans aucune construction de contexte progressive comme le fait Cascade. Contrairement à notre modèle qui utilise les contextes des décodeurs précédents, dans le modèle d'attention de base (Luong *et al.* (2015a)), l'encodeur crée le contexte pour prédire U_4 à partir de la concaténation des trois énoncés précédents. Les résultats montrent que Cascade a constamment surperformé le modèle de base de Seq2Seq.

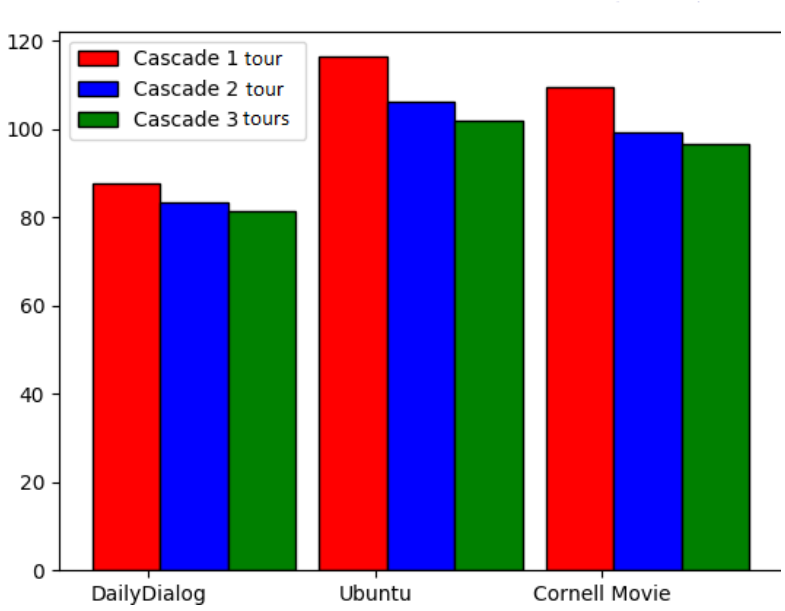


Figure 4.12: Résultats de perplexité pour le modèle Cascade par nombre de décodeurs utilisés et nombre de tours.

Enfin, afin d'étudier l'impact de l'ajout d'un historique de contexte dans chaque décodeur Cascade, nous avons évalué le modèle avec et sans historique de contexte. Cela a été fait en étudiant une version où l'attention du décodeur D est limitée au dernier état caché du décodeur C. Par conséquent, les équations d'attention de Cascade deviennent :

$$\begin{cases} Score\left(\overrightarrow{h}_{m,n}, \overline{h}_{m-1}\right) = \overrightarrow{h}^{T}_{m,n}\left[\overline{h}_{m-1}\right] \\ \overrightarrow{\alpha}_{n} = softmax\left(score(\overrightarrow{h}_{m,n}, \overline{h}_{m-1})\right) \\ \overrightarrow{c}_{m,n} = \overrightarrow{\alpha}^{T}_{n}[\overline{h}_{m-1}] \end{cases} \tag{4.1}$$

Où : $\overrightarrow{h}_{m,n}$ est l'état caché du décodeur m à la position n, et $\overline{h}_{m-1} = [\overrightarrow{h}_{m-1,1}; \cdots; \overrightarrow{h}_{m-1,s}]$ sont les états cachés du décodeur $m-1$. $\overrightarrow{\alpha}_{n}$ le vecteur des poids, et $\overrightarrow{c}_{m,n}$ est le vecteur de contexte du décodeur m à la position n, tel que défini dans la sous-section 3.4.2.

Comme illustré dans le tableau 4.8, le modèle Cascade sans historique de contexte a moins performé que le modèle utilisant l'historique montrant ainsi le mérite de l'approche Cascade.

4.5.2 Évaluation humaine

Le tableau 4.10 fournit des exemples des réponses générées qui ont été évaluées par les cinq évaluateurs humains. La première ligne montre des échantillons de la boîte de dialogue DailyDialog, la deuxième provient de Ubuntu et la dernière de Cornell Movie Dialog. Les exemples sont particulièrement instructifs sur l'intérêt d'utiliser l'historique des dialogues pour créer des contextes. Selon ces exemples, Cascade était le seul modèle à générer des réponses système plus longues pour les corpus Cornell Movie Dialog et Ubuntu, comme indiqué dans le tableau 4.19.

La moyenne de la longueur de la réponse pour les modèles entraînés par le grand corpus d'Ubuntu et un corpus de taille moyenne comme Cornel Movie, ont permit au modèle Cascade de générer des phrases plus longues par rapport aux autres modèles. Par exemple, le tableau 4.10 montre le dialogue *«Did you change your hair ?* $\rightarrow$ *No.* $\rightarrow$ *You might wanna think about it»* a permis au modèle Cascade de générer la réponse *«I like my appearance, I would not change.»*. Cette réponse est plus longue que toutes celles proposées par les autres modèles qui ont répondu en un ou deux mots, sauf pour VHRED où la réponse était *«i don't know.»*. Cependant, dans le petit corpus comme

	DailyDialog	Ubuntu	Cornell Movie
Transformer	7.42	8.87	5.81
HRED	7.56	8.81	5.92
VHRED	6.25	8.68	5.49
VHCR	6.31	8.12	5.62
HRAN	**7.46**	9.74	6.08
ReCoSa	7.41	9.78	6.11
Cascade	7.39	**9.86**	**6.13**

Tableau 4.9: Nombre moyen de mots générés à partir des ensembles de test pour chaque modèle.

DailyDialog, les modèles ReCoSa et HRAN génèrent des phrases plus longues avec une légère différence.

Le tableau 4.10 nous montre un exemple tiré de Dailydialog, où la boîte de dialogue *«can i help you , sir ?* → *yes , i'd like to withdraw some money .* → *fill in the slip , stating the exact amount you wish to withdraw , please .»* a permis au modèle Cascade de générer la réponse *«ok . here you are .»*. Cette réponse est plus courte que celles proposées par ReCoSa et HRAN. En revanche, pour le reste des modèles, leurs réponses étaient plus courtes que celle générée par notre modèle.

Le tableau 4.11 présente les résultats de l'évaluation humaine donnant les moyennes de victoire, de défaite et d'égalité pour chaque corpus et modèle. De toute évidence, Cascade a dépassé transformer de base, HRED, VHRED et VHCR par le nombre de victoires sur l'ensemble de données Ubuntu, et a égalé HRAN et ReCoSa compte tenu de leurs marges. D'un autre côté, il n'y avait pas de gagnant définitif pour les petits ensembles de données DailyDialog et Cornell Movie, car les résultats entre la moyenne, la défaite et la victoire étaient très proches avec un grand nombre des cas d'égalité. Ainsi, il semble que la taille de l'ensemble de données soit importante pour la distinction des modèles. Au cours de nos différentes expériences, nous avons remarqué que si la conversation dure plus d'un tour, les performances et la supériorité potentielle

DailyDialog	**Context**	can i help you , sir ? → *yes , i'd like to withdraw some money .* → fill in the slip , stating the exact amount you wish to withdraw , please .
	Réponse	
	Transformer	i think i will .
	HRED	ok . .
	VHRED	here you .
	VHCR	ok . <unk> .
	HRAN	ok . is that all right ? .
	ReCoSa	ok . here's my bank book . . .
	Cascade	ok . here you are .
	Référence	here you are .
Ubuntu	**Context**	i just added one usb printer but not sure what the uri should be→ *firefox localhost :8080* → firefox ?
	Réponse	
	Transformer	it's a free
	HRED	http ://tldp.org <org> abs guide
	VHRED	if you want to , i would recommend it
	VHCR	i do n't know what you mean by that.
	ReCoSa	firefox local local local.
	HRAN	try to use the default browser in firefox
	Cascade	yes firefox localhost :8080 firefox http ://localhost :8080 has a web based interface.
	Référence	http : //home in your browser
Cornell Movie	**Contexte**	Did you change your hair ? → *No.* → You might wanna think about it
	Réponse	
	Transformer	no . . . .
	HRED	thanks .
	VHRED	i do n't know .
	VHCR	ok .
	HRAN	thanks a . .
	ReCoSa	No .
	Cascade	i like my appearance, i would not change.
	Référence	I want to keep my hair long.

Tableau 4.10: Exemples de réponses générées par les différents modèles pour les trois corpus considérés.

de Cascade par rapport aux autres modèles augmentent. Les résultats de l'évaluation automatique mènent à la même conclusion.

4.5.3 Discussion

Les évaluations automatiques et humaines sont toutes deux d'accord avec l'hypothèse suivante : une architecture utilisant une attention multidimensionnelle pour construire le contexte fournit des performances de génération des réponses égales ou meilleures

	Modèle	Victoire	Défaite	Égalité
Ubuntu	Cascade vs Transformer	**51.3**±3.4	43.6±4.1	5.1±3.2
	Cascade vs HRED	**52.4**±2.3	41.7±2.7	5.9±1.8
	Cascade vs VHRED	**52.4**±2.3	41.7±2.7	5.9±1.8
	Cascade vs VHCR	**53.7**±2.1	40.6±2.1	5.7±2.9
	Cascade vs HRAN	**41.9**±3.1	41.4±3.2	16.7±4.3
	Cascade vs ReCoSa	39.8±3.4	**42.7**±3.2	17.4±4.1
DailyDialog	Cascade vs Transformer	**36.2**±4.1	29.1±3.2	34.7±4.2
	Cascade vs HRED	**32.5**±1.7	32.2±2.1	35.3±2.3
	Cascade vs VHRED	**29.7**±1.3	29.0±1.1	41.3±2.7
	Cascade vs VHCR	**28.3**±1.6	28.2±1.9	43.5±2.1
	Cascade vs HRAN	**32.4**±2.9	31.2±3.7	36.3±3.7
	Cascade vs ReCoSa	**31.3**±3.1	29.1±2.8	39.6±3.2
Cornell Movie	Cascade vs Transformer	**36.8**±3.1	28.4±4.1	34.7±3.9
	Cascade vs HRED	**33.5**±1.4	33.2±2.9	33.3±2.3
	Cascade vs VHRED	**28.5**±2.8	28.2±1.8	43.3±1.5
	Cascade vs VHCR	**28.4**±1.3	28.2±2.2	43.4±2.4
	Cascade vs HRAN	**31.7**±3.1	30.1±3.4	38.2±4.1
	Cascade vs ReCoSa	**31.4**±2.8	29.3±3.7	39.3±4.2

Tableau 4.11: La moyenne des scores de victoire, de défaite et de l'égalité de l'évaluation humaine.

que celle adoptant une attention régulière ou une auto-attention en général. Cela peut s'expliquer par le fait que l'attention multidimensionnelle proposée tient compte de toutes les dépendances possibles lors de la recherche de relations entre les mots sources et les mots cibles.

En revanche, Le modèle est fourni avec un grand nombre de paramètres utilisant 3 décodeurs. De plus, il utilise des unités RNRs (GRU). Cela permet au modèle d'hériter de toutes les limites du RNR, notamment avec la naissance d'autres architectures basées sur l'auto-attention comme les transformers, qui ont montré de bons résultats dans la tâche de traduction automatique. Le modèle ReCoSa a réussi à battre notre modèle en perplexité avec le grand corpus de Ubuntu. Bien que ReCoSa utilise toujours RNR, une

simple introduction de l'auto-attention dans ReCoSa a permit de réduire la perplexité.

Afin de résoudre ces limites, nous avons introduit une nouvelle architecture qui peut apporter des corrections architecturales dans la prochaine section.

4.5.4 Attention multidimensionnelle

La figure 4.13 montre des cartes thermiques des attentions des décodeurs *B*, *C* et *D*. Cette figure révèle les mots sources les plus significatifs derrière chaque mot cible généré. Par exemple dans la dernière carte thermique, le mot «*I*» est responsable de la génération du mot «*You*», qui pourrait être interprété comme chaque fois que l'interlocuteur dit «*I*», le décodeur essaie de répondre par «*You*».

Chacune des cartes thermiques a un axe horizontal qui représente une sortie de décodeur et un axe vertical qui représente la sortie des décodeurs ou de l'encodeur précédent. Les couleurs des pixels représentent l'attention entre eux telle que fournie par l'équation 3.16. La première carte concerne le décodeur *B* dans l'axe horizontal et l'encodeur *A* dans l'axe vertical. La deuxième est la sortie du décodeur *C* avec le décodeur *B* (en haut) et l'encodeur *A* (en bas). La troisième correspond à la sortie du décodeur *D* dans l'axe horizontal avec le décodeur *C* (haut), décodeur *B* (milieu) et l'encodeur *A* (bas).

Dans l'exemple donné, nous voyons que l'attention devient plus riche au fur et à mesure que l'on passe du décodeur *B* au décodeur *D*, puisque le nombre de pixels jaunes augmente de la première carte thermique à la dernière. En particulier, la première requête E_1 de l'utilisateur ("*excuse me, professor ...*") crée une attention qui est reliée avec la réponse de l'agent E_2. Puis, la réponse de l'agent E_2 ("*Is this something...*") et le contexte de la requête E_1 créent une nouvelle attention qui est reliée avec la requête E_3 posée par l'utilisateur ("*no, i have to do this ...*"). Enfin, la réponse de l'agent E_4 ("*Have you arranged ...*") avec les contextes de E_2 et E_1 ont créé une dernière attention avec la requête E_3.

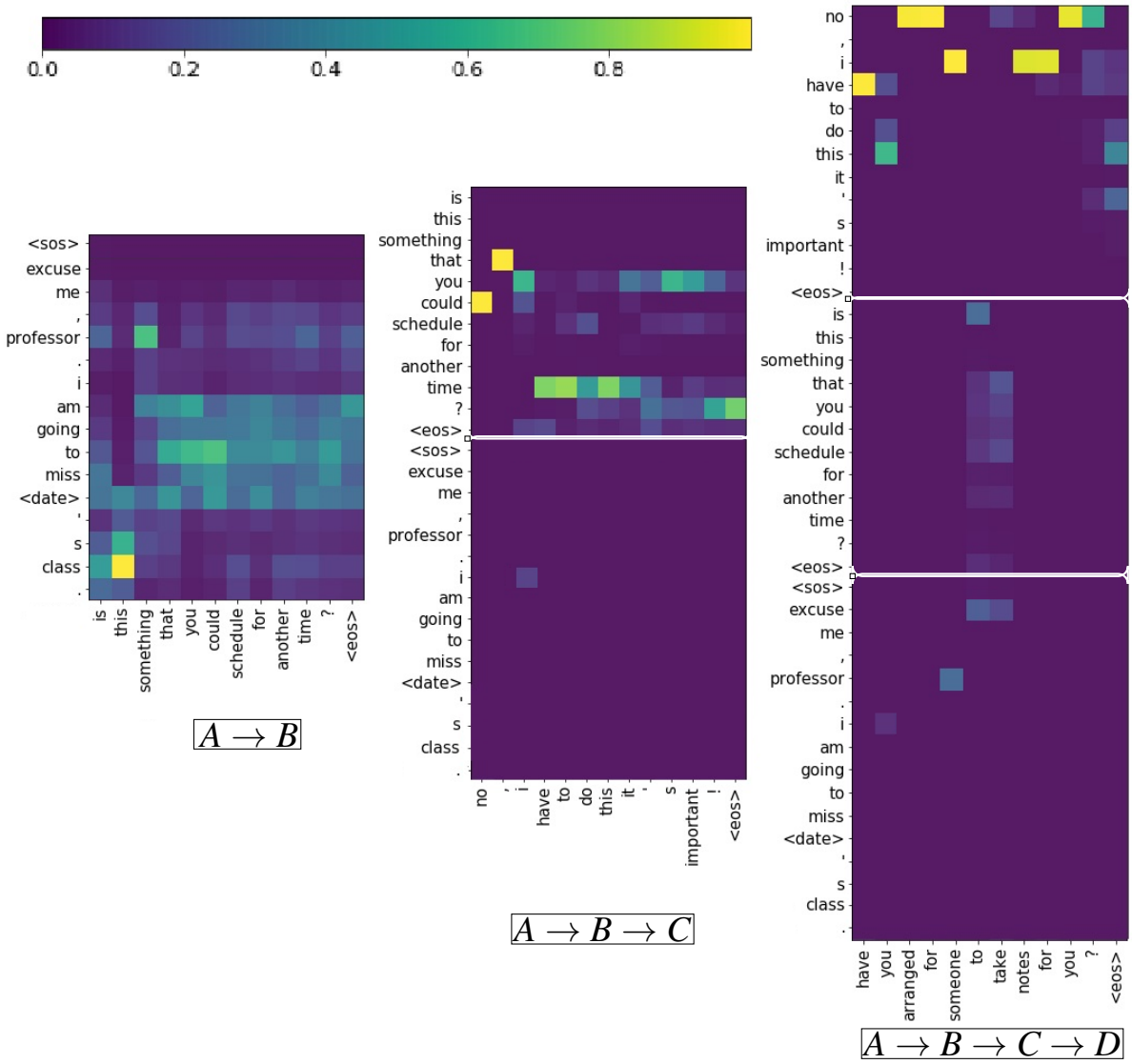


Figure 4.13: Visualisation de l'évolution de l'attention au fur et à mesure de sa progression de l'encodeur A au décodeur D pour un exemple de dialogue.

Notez que les cartes d'attentions hiérarchiques produites peuvent également servir à expliquer les réponses erronées de l'agent lorsqu'une entité nommée par l'utilisateur

est absente du corpus d'apprentissage, en montrant l'attention créée avec chaque mot de la requête et sa propagation à travers les décodeurs.

4.5.5 Analyse des erreurs

Cascade peut également générer des réponses ambiguës ou erronées. Par exemple, étant donné la séquence *"hello how are you –> good .... –> where is your last trip"*, ça répond avec *"usa... china.. I do not understand"*. Cette mauvaise prédiction peut être dûe à un entraînement insuffisant ou à des données déséquilibrées, car nous avons remarqué que les deux noms de pays sont prédominants dans le corpus de l'entraînement. La réponse générée agrège également les noms de pays avec des signes de ponctuation comme s'ils constituaient des entités nommées uniques. Enfin, il y a parfois confusion entre les verbes *to be* et *to have* dans la réponse. Ces problèmes ne semblent pas avoir été résolus auparavant dans les modèles de comparaison (Sordoni *et al.*, 2015b; Serban *et al.*, 2017c; Park *et al.*, 2018; Xing *et al.*, 2018; Zhang *et al.*, 2019a), et méritent d'être approfondis. En particulier, il peut être utile d'augmenter les données d'entraînement avec des connaissances externes issues de bases lexicales, de lexiques ou d'autres bases de connaissances telles les ontologies comme *BabelNet* (Navigli et Ponzetto, 2012).

Les erreurs de prétraitement peuvent également influencer les résultats. Par exemple, les noms de personnes ne sont pas reconnus et modifiés en entité nommée <personne> s'ils sont mal orthographiés pour commencer par une lettre minuscule. Dans la même veine, des mots comme « *100 :yuans* » sont considérés comme des mots rares et remplacés par <UNK> (les mots hors vocabulaire), car ils ne sont pas reconnus comme le type d'entité <Monnaie>. Ces erreurs de prétraitement sont dues à l'analyseur utilisé *Spacy* qui ignore les entités nommées commençant par une lettre minuscule. Dans de tels cas, la mauvaise réponse peut être générée. Cependant, ce comportement de prétraitement est dû à la limitation des analyseurs d'entités nommées (Fotsoh, 2018) et non aux architectures des modèles. Ce problème est un sujet qui mérite d'être étudié dans

le futur.

4.6 Évaluation du modèle de Transformer basé sur l'attention multidimensionnelle (MDA)

De la même manière, nous avons utilisé les trois ensembles de données, ainsi que les mêmes mesures de performance afin d'évaluer le modèle en ajoutant une comparaison basée sur un corpus de traduction. Nous avons également mené des expériences avec les cinq modèles hiérarchiques (HRED, VHRED, VHCR, HRAN et ReCoSa) ainsi que le transformer comme modèle de base.

Nous avons utilisé les mêmes ensembles de données et mesures de performance comme décrit ci-après.

4.6.1 Résultats de l'évaluation automatique

Le tableau 4.12 montre les valeurs de la similitude cosinus entre la réponse réelle et la réponse générée, en utilisant les métriques basées sur les plongements des mots moyens, gourmands, les vecteurs extrêmes ainsi que la mesure de perplexité pour les différents modèles. Dans chaque cas, une seule réponse est générée pour un contexte donné. Le Transformer avec MDA obtient les meilleurs résultats suivant toutes les métriques basées sur les plongements des mots, suivi par ReCoSa et HRAN.

Le résultat des métriques de plongements du Transformer MDA est meilleur que ceux des autres modèles. Ces métriques (Liu *et al.*, 2016) cartographient les réponses dans l'espace vectoriel et calculent la similitude cosinus (Rus et Lintean, 2012a). Elles nous informent sur la position des vecteurs de réponse générée par rapport à la réponse réelle ; plus la valeur du cosinus est proche de 1, plus les deux réponses sont identiques. L'amélioration de la métrique gourmande a démontré que les mots générés sont proches d'un mot dans la réponse réelle et que l'inverse est également vrai. La métrique moyenne prend en compte tous les mots et toutes les dimensions afin de comparer

deux vecteurs de réponses. Pour s'assurer que l'amélioration du résultat de la métrique moyenne ne se fait pas au prix d'une augmentation du nombre de mots vides ou non pertinents, on la vérifie en utilisant le vecteur extrême. Ce vecteur ignore les mots non pertinents (les vecteurs de plongement des mots fréquents dans le corpus, se trouvent proches de l'origine dans l'espace vectoriel (Liu *et al.*, 2016)).

Le modèle proposé présente également la plus faible perplexité. Il montre qu'un modèle de conversation à un seul encodeur et un seul décodeur, par opposition à celui utilisant un traitement hiérarchique ou des modules supplémentaires, conduit à des résultats moins fluides.

Notre modèle MDA entraîné par le corpus Ubuntu a obtenu la perplexité la plus faible de 87,41 contre un score de 97,16 obtenu par ReCoSa avec une différence de 9,95. Notons que ReCoSa a obtenu une perplexité faible par rapport aux autres modèles hiérarchiques. Le corpus Ubuntu se distingue par sa taille de 892 000 conversations.

La deuxième plus grande marge de différence de perplexité est obtenue par l'entraînement de notre modèle avec DailyDialog. Le modèle a obtenu un score de 79,62 contre 84,98 obtenu par ReCoSa, avec une différence de 5,36. Ce corpus est caractérisé par la qualité du texte écrit (Li *et al.*, 2017b). Cependant, sa taille de 11118 conversations est plus petite que celles de Ubuntu et Cornell Movie,.

Le troisième résultat obtenu par notre modèle entraîné par Cornell Movie est de 95,57 contre 96,82 avec une différence de 1,25. Ce corpus contient des données extraites des sous-titrages des films. Ainsi, il contient 79879 conversations, avec beaucoup de bruit et des phrases mal construites.

Par définition, la perplexité est l'inverse de la probabilité d'observer les mots du texte conjointement afin de former les phrases par un modèle de langue (Pietquin et Hastie, 2013a). Ce qui signifie qu'il a une bonne compréhension du fonctionnement de la

langue. En conséquence, notre modèle a appris davantage des séquences des motifs et des segments de texte afin de générer des réponses meilleures par rapport aux autres modèles.

Corpus	**DailyDialog**				**Ubuntu**				**Cornell Movie**			
Modèle	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL	Moyenne	Gourmand	Extrêmes	PPL
Transformer	0.514	0.372	0.311	83.41	0.438	0.387	0.291	103.08	0.407	0.387	0.313	98.38
HRED	0.636	0.459	0.391	85.08	0.638	0.456	0.378	121.08	0.569	0.415	0.356	115.01
VHRED	0.620	0.432	0.363	96.73	0.549	0.400	0.308	192.93	0.557	0.402	0.352	186.79
VHCR	0.624	0.439	0.352	83.98	0.565	0.416	0.328	112.18	0.548	0.393	0.342	104.90
HRAN	0.611	0.419	0.314	83.48	0.568	0.429	0.364	101.44	0.512	0.387	0.332	97.41
ReCoSa	0.626	0.421	0.321	84.98	0.573	0.411	0.351	97.16	0.537	0.396	0.341	96.82
MDA	**0.671**	**0.524**	**0.398**	**79.62**	**0.643**	**0.512**	**0.421**	**87.41**	**0.653**	**0.512**	**0.442**	**95.57**

Tableau 4.12: Comparaison des modèles sur trois corpus d'entraînement pour la tâche de dialogue, en utilisant les trois métriques de plongements décrits et la perplexité

4.6.2 Résultats de l'évaluation humaine

Le tableau 4.13 fournit des exemples des réponses générées qui ont été évaluées par les cinq évaluateurs humains. La première ligne montre un échantillon d'une séquence des énoncés du corpus DailyDialog représentant un état de contexte de la conversation. La deuxième provient d'un état de conversation Ubuntu et la dernière de Cornell Movie. Le tableau 4.14 résume les résultats de l'évaluation humaine fournissant les moyennes des victoires, des défaites et des égalités pour chaque corpus et modèle. Ces valeurs obtenues confirment celles de l'évaluation automatique sur les meilleures performances obtenues du modèle proposé avec l'architecture MDA Transformer.

Nous remarquons également que plus l'ensemble d'entraînement est grand, meilleurs seront les résultats. Ainsi, il semble que la taille de l'ensemble de données soit importante pour la distinction des modèles. Le tableau 4.15 montre le nombre moyen de mots générés à partir du corpus de test pour chaque modèle. On peut remarquer que le modèle MDA génère des réponses plus longues par rapport aux autres modèles. Cependant, le

DailyDialog	**Context**	i ' m fine . thank you . → *how may i help you ?* → i need to make a deposit . → *will you be depositing cash or a check ?* → i will be depositing cash .
	Réponse	
	Transformer	i think i will .
	HRED	here you are .
	VHRED	four <unk> .
	VHCR	is that all ?
	HRAN	yes , i am . .
	ReCoSa	ok ! thank you .
	Cascade	how much you want to deposit .
	MDA	how much would you like to deposit ?
	Référence	How much do you want to deposit ?
Ubuntu	**Context**	i just added one usb printer but not sure what the uri should be→ *firefox localhost :8080* → firefox ?
	Réponse	
	Transformer	it's a free
	HRED	http ://tldp.org <org> abs guide
	VHRED	if you want to , i would recommend it
	VHCR	i do n't know what you mean by that.
	ReCoSa	firefox local local local.
	HRAN	try to use the default browser in firefox
	Cascade	yes firefox localhost :8080 firefox http ://localhost :8080 has a web based interface.
	MDA	firefox local browser.
	Référence	http : //home in your browser
Cornell Movie	**Contexte**	Did you change your hair ? → *No.* → You might wanna think about it
	Réponse	
	Transformer	no . . . .
	HRED	thanks .
	VHRED	i do n't know .
	VHCR	ok .
	HRAN	thanks a . .
	ReCoSa	No .
	Cascade	i like my appearance, i would not change.
	MDA	i might change my mind . . .
	Référence	I want to keep my hair long .

Tableau 4.13: Exemples de réponses générées par les différents modèles pour les trois corpus considérés.

	Modèle	Victoire	Défaite	Égalité
Ubuntu	MDA vs Transformer	**43.3**±2.6	41.2±2.8	15.6±3.4
	MDA vs HRED	**49.9**±2.4	42.9±2.8	6.2±2.9
	MDA vs VHRED	**57.1**±2.7	38.6±2.6	3.6±1.7
	MDA vs VHCR	**53.8**±2.7	41.5±2.5	4.6±1.8
	MDA vs HRAN	**42.4**±2.1	41.2±2.6	16.7±2.4
	MDA vs Cascade	**49.8**±3.7	42.7±3.4	7.4±3.7
	MDA vs ReCoSa	**48.9**±2.1	44.8±2.1	5.3±1.8
DailyDialog	MDA vs Transformer	**33.1**±2.7	26.8±2.9	41.1±3.1
	MDA vs HRED	**34.4**±1.7	31.3±2.1	34.3±2.2
	MDA vs VHRED	**31.6**±1.3	28.9±1.1	39.3±3.6
	MDA vs VHCR	**30.3**±2.7	29.3±2.8	40.4±3.3
	MDA vs HRAN	**31.2**±2.3	28.9±2.1	41.1±3.1
	MDA vs Cascade	**33.2**±2.8	27.8±3.1	39.1±3.7
	MDA vs ReCoSa	**30.4**±1.6	27.0±1.9	42.6±3.2
Cornell Movie	MDA vs Transformer	**34.1**±3.1	29.9±2.8	36.1±3.3
	MDA vs HRED	**32.5**±2.3	31.2±1.8	36.3±3.4
	MDA vs VHRED	**31.4**±1.9	26.2±1.7	42.3±2.6
	MDA vs VHCR	**30.5**±2.7	28.1±3.4	41.4±3.1
	MDA vs HRAN	**30.6**±3.3	28.3±3.2	41.9±3.4
	MDA vs Cascade	**32.3**±3.3	28.2±3.1	39.7±3.5
	MDA vs ReCoSa	**29.3**±3.2	27.4±3.1	43.3±3.3

Tableau 4.14: Moyennes des scores des victoires, défaites et égalités de l'évaluation humaine.

modèle Cascade génère de longues phrases lorsque le corpus est de grande taille comme celui d'Ubuntu, mais ne dépasse pas la moyenne des longueurs des phrases générées par Transformer MDA.

4.6.3 Analyse des erreurs

Notre modèle proposé peut également générer des réponses ambiguës ou grammaticalement erronées. Par exemple, dans le tableau 4.13 étant donné la séquence "(i just added one usb printer but not sure what the uri should be→ *firefox localhost :8080* → fi-

	DailyDialog	Ubuntu	Cornell Movie
Transformer	7.42	8.87	5.81
HRED	7.56	8.81	5.92
VHRED	6.25	8.68	5.49
VHCR	6.31	8.12	5.62
HRAN	7.46	9.74	6.08
ReCoSa	7.41	9.78	6.11
Cascade	7.39	9.86	6.13
MDA	**7.87**	**9.89**	**6.43**

Tableau 4.15: Nombre moyen de mots générés à partir des ensembles de test pour chaque modèle.

refox ?)" dans le corpus Ubuntu, ça répond avec "firefox local browser." Bien que cette réponse compte des mots informatifs ayant une relation directe avec le contexte, elle ne constitue pas une phrase complète, puisqu'elle ne contient aucun verbe. Cette mauvaise réponse pourrait être due à la nature des données hautement bruitées des réseaux sociaux du système Ubuntu d'où ce corpus est extrait. Nous notons que ce problème n'est pas exclusif à notre modèle, puisque tous les autres modèles proposent des réponses avec des phrases incomplètes et mal construites.

Un autre exemple, dans le tableau 4.13 étant donné la séquence "*Did you change your hair ?* $\rightarrow$ *No.* $\rightarrow$ *You might wanna think about it*" dans notre modèle entraîné par le corpus Cornell Movie, ça répond avec "might change my mind . . .". Bien que cette réponse soit correcte, la phrase contient des répétitions du point à la fin de la phrase. Les doublons ont été remarqués dans la génération de réponse produite par tous les modèles et dans différents exemples. comme HRAN a répondu par '*thanks a . .*' et Transformer a répondu par '*no . . . .*'.

Un autre exemple, dans le tableau 4.13 étant donné la séquence "*Did you change your hair ?* $\rightarrow$ *No.* $\rightarrow$ *You might wanna think about it*" dans notre modèle, qui a été entraîné sur le corpus DailyDialog ; la réponse générée est "*how much would you like to*

deposit ?". Cette réponse est correcte et dans le contexte de la conversation. De plus, la réponse de notre modèle partage 6 mots avec la réponse de la référence. Alors que Cascade a bien répondu à la conversation avec sa réponse de "*how much you want to deposit*", et la réponse générée partage le même nombre de mots avec la référence. La différence entre les deux réponses est que notre modèle reconnaît que la question doit se terminer par un point d'interrogation contrairement au modèle Cascade qui termine la question par un point.

4.6.4 Résultats de l'attention multidimensionnelle

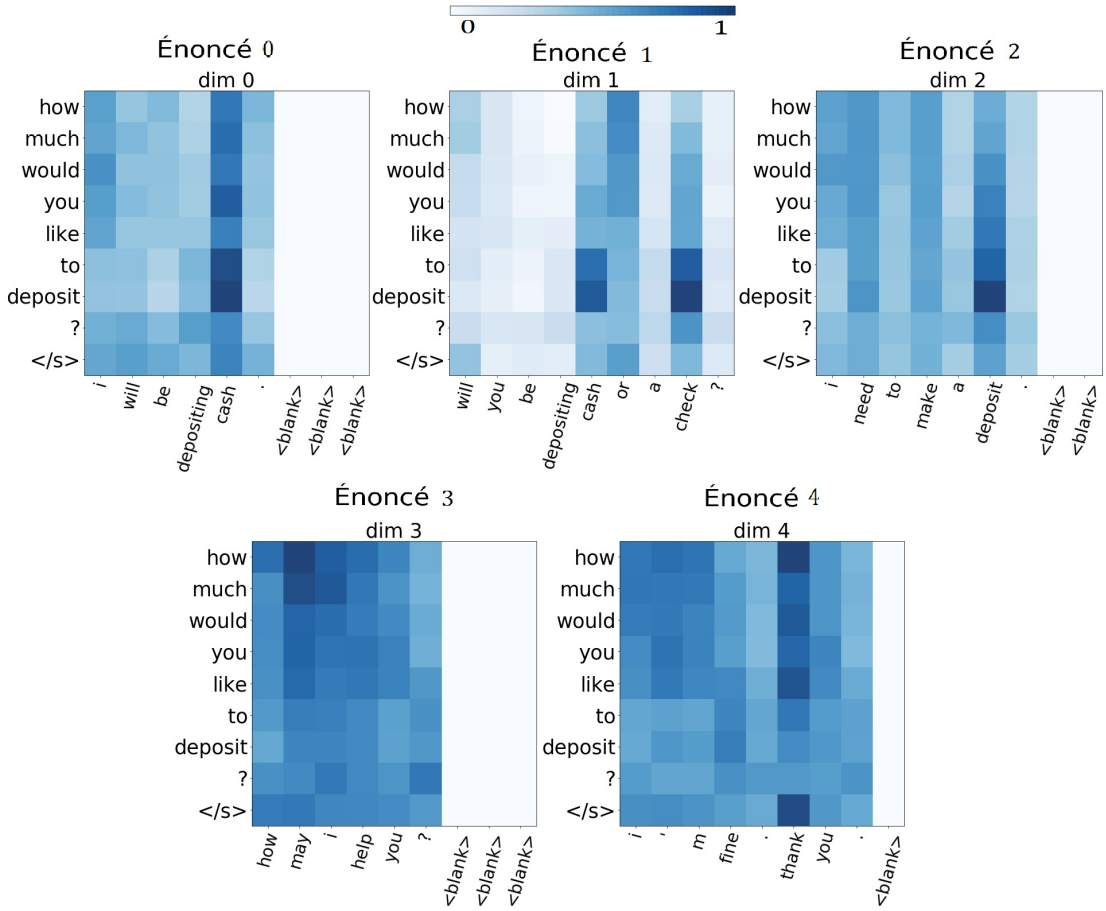


Figure 4.14: Visualisation détaillée de l'évolution de l'attention pour un exemple de dialogue.

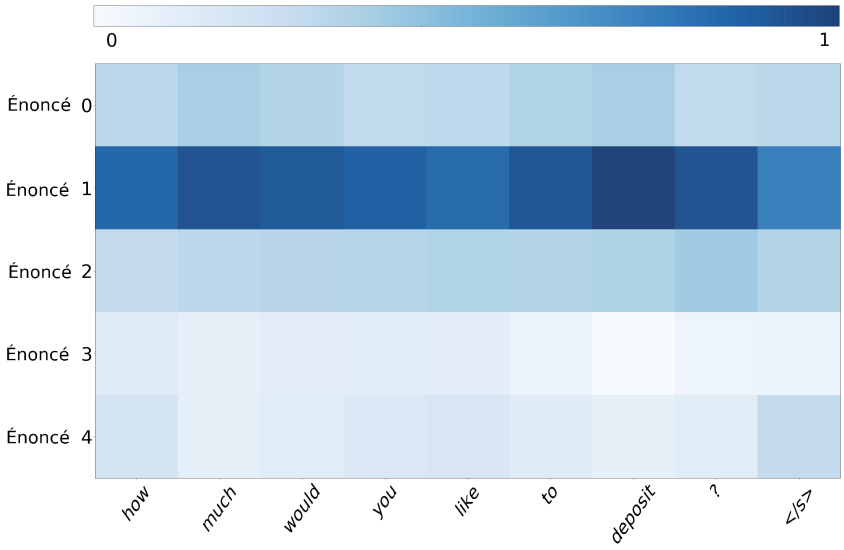


Figure 4.15: Visualisation de l'évolution de l'attention au niveau de l'énoncé pour un exemple de dialogue.

La figure 4.14 visualise l'attention multidimensionnelle pour un exemple de DailyDialog dans le tableau 4.13. Tous les axes verticaux des cartes thermiques illustrées affichent la réponse générée, les axes horizontaux affichent les énoncés de dialogue successifs et les couleurs de pixels représentent l'attention, calculée par l'équation 3.21, entre chaque paire de mots (mot source / mot de sortie). La première carte concerne l'énoncé 0, la deuxième l'énoncé 1 et ainsi de suite. La figure 4.15 fournit une visualisation alternative où l'axe horizontal est la sortie du décodeur et l'axe vertical représente la somme des attentions de mots groupés pour chaque énoncé.

Dans cet exemple, nous voyons à partir de la figure 4.15 que le mot « deposit » dans la réponse était principalement influencé par l'énoncé 1, avec une aide des énoncés 0 et 2.

La figure 4.14 va plus loin en montrant qu'il était les mots « cash » et « check » de l'énoncé 1, le mot «cash» de nouveau de l'énoncé 0, et les mots « to » et « deposit » de l'énoncé 2, qui ont le plus contribué au « deposit ». La figure 4.15 est le résultat de la sommation de l'attention MDA de la figure 4.14. Chaque matrice a été regroupée

par une sommation de l'axe des mots d'entrées dans un vecteur, l'une après l'autre créant ainsi la figure 4.14. Cette sommation est un résumé qui nous permet d'analyser les énoncés qui ont le mieux contribué à la génération de mots. La figure 4.16 nous montre la somme effectuée sur le tenseur d'attention en regroupant les *M* mots d'entrée. L'objectif final est d'obtenir une matrice d'attention de la figure 4.15.

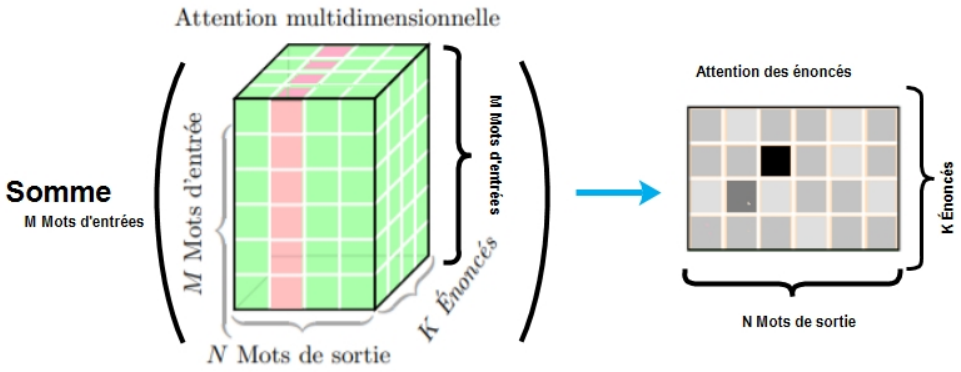


Figure 4.16: Regroupement de l'attention 3D par la somme de l'axe des mots d'entrée afin d'obtenir l'attention 2D.

Il est à noter également que l'énoncé le plus important pour contribuer à la réponse, c'est l'énoncé 1 qui provenait de notre modèle proposé. Cela confirme l'hypothèse selon laquelle les historiques des requêtes et des réponses sont importants afin de concevoir un processus de génération des conversations efficace.

4.6.5 Score BLEU pour la modélisation de conversation

Cette métrique d'évaluation automatique est couramment utilisée pour les tâches de traduction automatique et de synthèse de texte. Les références de vérité terrain sont généralement bien définies et permettent ainsi des comparaisons des sorties relativement simples (Papineni *et al.*, 2002). La situation est plus ambiguë dans les systèmes de dialogue à domaine ouvert, où les réponses générées sont souvent mal corrélées avec le jugement humain en raison de la variété typique des choix possibles des réponses (Liu

et al., 2016).

Pourtant, il existe des cas où la métrique *BLEU* peut être pertinente, comme dans les dialogues avec des questions fermées ou des réponses prédéterminées qui existent déjà dans les corpus. Les discussions fermées qui peuvent se terminer avec un remerciement (ex : *Q : here's the cash for you . R : thanks ..*), ou une affirmation ,*Q : Can i help you , sir ?, R : yes ,* ou encore des énumérations des réponses possibles (comme : *Q : how old is the queen ?,R : [83,.. ,i don't know]*).

Le tableau 4.16 montre les scores *BLEU* obtenus par les différents modèles pour les corpus DailyDialog, Ubuntu et Movis-DIC. Les valeurs sont petites comme prévu pour ces corpus à domaine ouvert. Cependant, notre modèle surpasse les autres modèles de l'état de l'art cités, en matière de performance. Cela est éventuellement dû à sa génération de réponses plus concises comme l'indiquent ses scores de la faible perplexité dans le tableau 4.12.

	DailyDialog	**Ubuntu**	**Cornell Movie**
Transformer	0.42	0.87	0.21
HRED	0.56	1.41	0.47
VHRED	1.25	1.68	0.49
VHCR	1.31	2.12	0.62
HRAN	1.46	2.74	1.08
Cascade	1.39	1.89	0.48
ReCoSa	1.72	2.98	1.17
MDA	**2.87**	**3.29**	**1.43**

Tableau 4.16: Score *BLEU* du MDA par rapport aux autres modèles

4.6.6 Test de signification statistique

Nous avons utilisé le rééchantillonnage bootstrap (Koehn, 2004) pour évaluer la signification statistique de nos comparaisons. La signification statistique représente une étude de stabilité du score *BLEU* et la *Perplexité* pour les résultats retrouvés. L'utilisation d'un rééchantillonnage aléatoire avec remplacement a permis de créer 100 ensembles de tests ayant la moitié de la taille de chacun des ensembles de tests DailyDialog, Ubuntu et Cornell Movie. Ensuite, les scores *BLEU* et la *Perplexité* pour chaque ensemble de tests ont été calculés par les différents modèles.

Tout d'abord, nous analysons la signification statistique du résultat en score *BLEU* puis la *Perplexité* de l'ensemble de tests Daily Dialog.

La figure 4.17 fournit les graphes des scores *BLEU* obtenus et leurs valeurs moyennes pour chaque modèle. Les résultats montrent un intervalle de confiance à 95 % de [2.0, 3.4], [0.3, 0.7], [0.5, 0.8], [1.1, 2.1], [0.7, 1.9], [0.7, 1.8] et [1.1, 2.2] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa respectivement, avec des valeurs moyennes de 2.77, 0.5, 0.7, 1.4, 1.2, 1.2 et 1.7. Ces valeurs sont très proches de celles rapportées dans le tableau 4.16 pour l'ensemble de tests DailyDialog : 2.87. De plus, la figure 4.18 montre une différence de score *BLEU* constamment positive entre notre modèle et les autres.

La figure 4.19 montre les graphes des valeurs de la *Perplexité* obtenus et leurs valeurs moyennes pour chaque modèle. Les résultats montrent un intervalle de confiance à 95 % de [71.2, 84.4], [74.1, 86.2], [83.7, 93.9], [78.1, 90.2], [84.9, 103.3], [72.7, 92.6] et de [77.9, 91.3] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa respectivement, avec des valeurs moyennes de 78.2, 81.0, 88.8, 85.1, 97.5, 82.7 et 84.9. Ces valeurs sont très proches de celles rapportées dans le tableau 4.12 pour l'ensemble de tests DailyDialog : 79.62. De plus, la figure 4.20

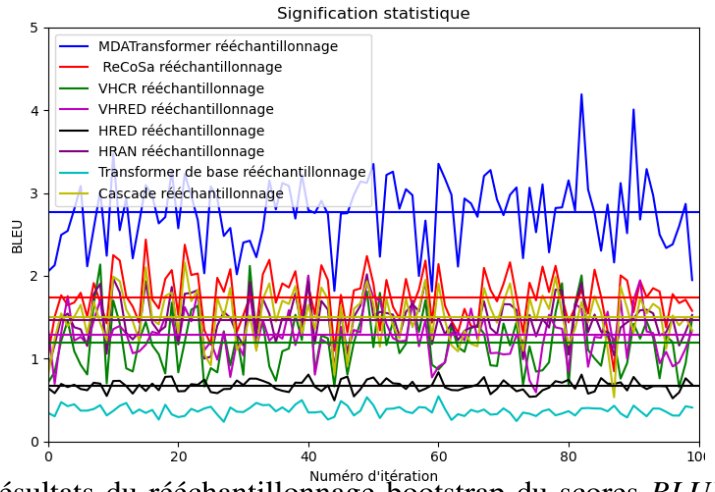


Figure 4.17: Résultats du rééchantillonnage bootstrap du scores *BLUE* pour 100 ensembles de tests dérivés de DailyDialog.

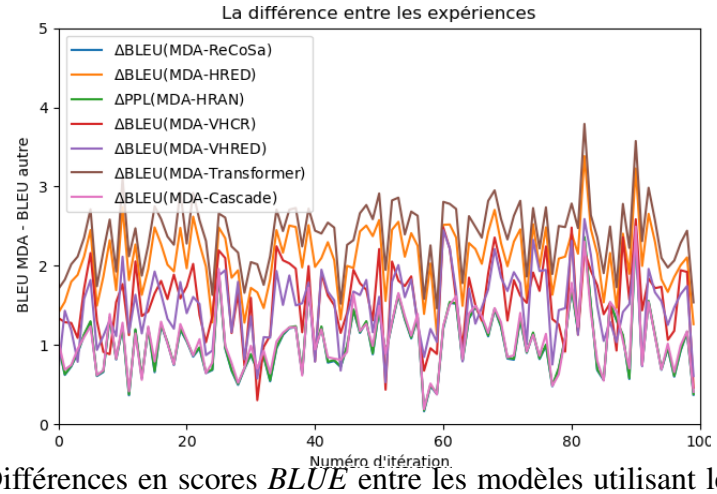


Figure 4.18: Différences en scores *BLUE* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés de DailyDialog.

montre une différence négative de *Perplexité* dans plus de 95% de rééchantillonnage utilisé entre notre modèle et les autres. Ceci confirme que notre modèle donne des valeurs de perplexité plus faibles pour l'ensemble de tests DailyDialog.

Ensuite, nous avons analysé la signification statistique du résultat en score *BLEU* puis la *Perplexité* de l'ensemble de test Ubuntu.

La figure 4.21 fournit les graphes des scores *BLEU* obtenus en utilisant des tests dérivés du corpus Ubuntu et leurs valeurs moyennes pour chaque modèle. Les résultats

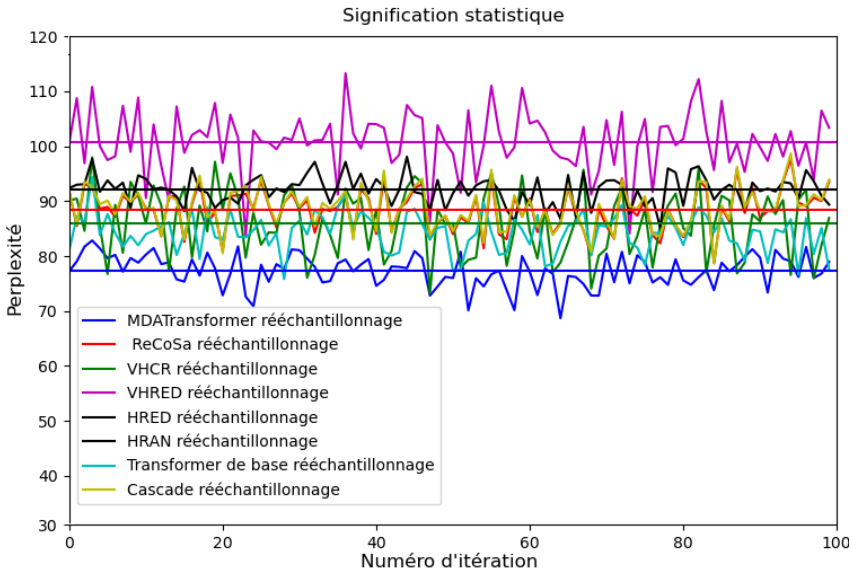


Figure 4.19: Résultats du rééchantillonnage bootstrap du scores de la *Perplexité* pour 100 ensembles de tests dérivés de DailyDialog.

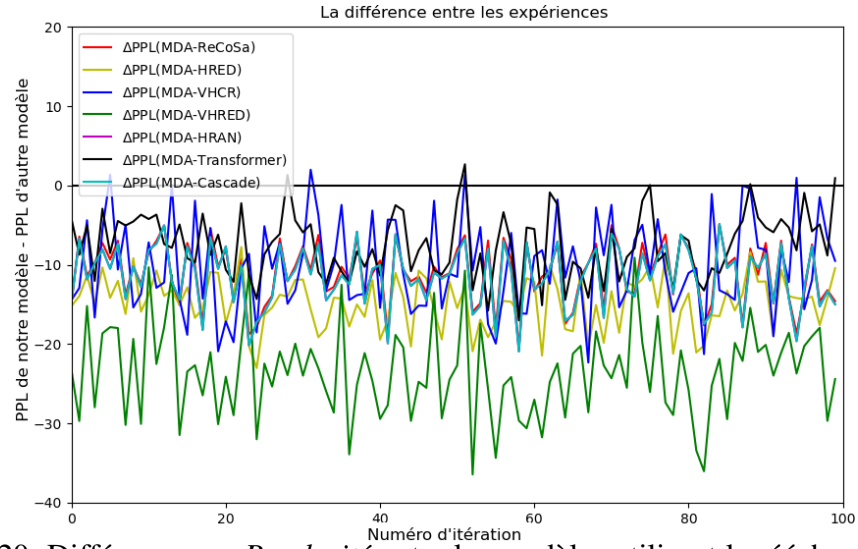


Figure 4.20: Différences en *Perplexité* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés de DailyDialog.

montrent un intervalle de confiance à 95 % de [2.9, 3.6], [0.7, 1.1], [1.3, 1.7], [2.1, 2.9], [1.5, 1.9], [1.8, 2.3] et [2.6, 3.2] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa respectivement avec des valeurs moyennes de 3.2, 0.7 , 1.5, 1.7, 1.7, 2.1, et 2.9. Ces valeurs sont très proches de celles rapportées dans le tableau 4.16 pour l'ensemble de tests Ubuntu : 3.29. De plus, la figure 4.22

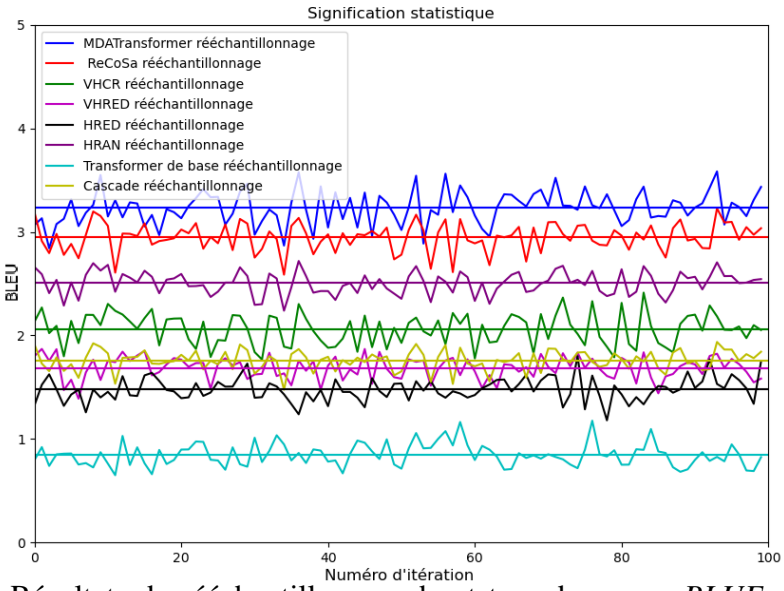


Figure 4.21: Résultats du rééchantillonnage bootstrap du scores *BLUE* pour 100 ensembles de tests dérivés de corpus Ubuntu.

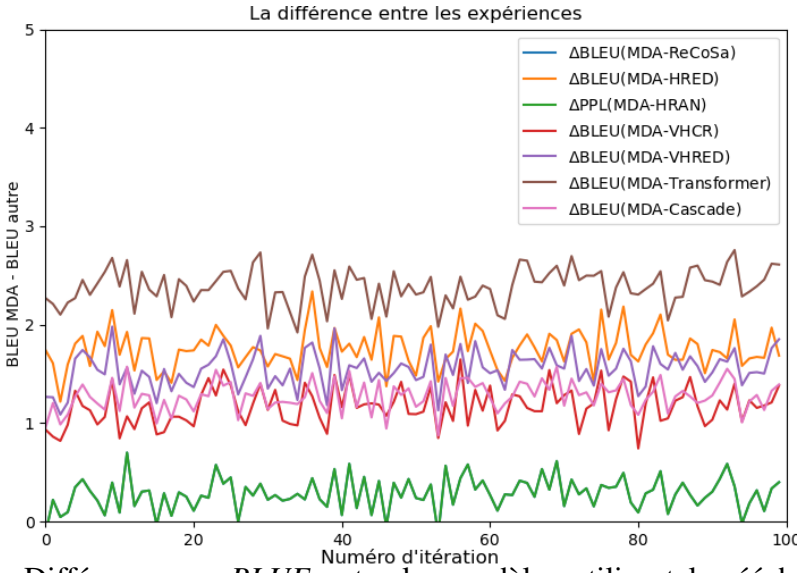


Figure 4.22: Différences en *BLUE* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés de corpus Ubuntu.

montre une différence de score *BLEU* constamment positive entre notre modèle et les autres.

La figure 4.23 fournit les graphes des valeurs de la *perplexité* obtenus en utilisant des

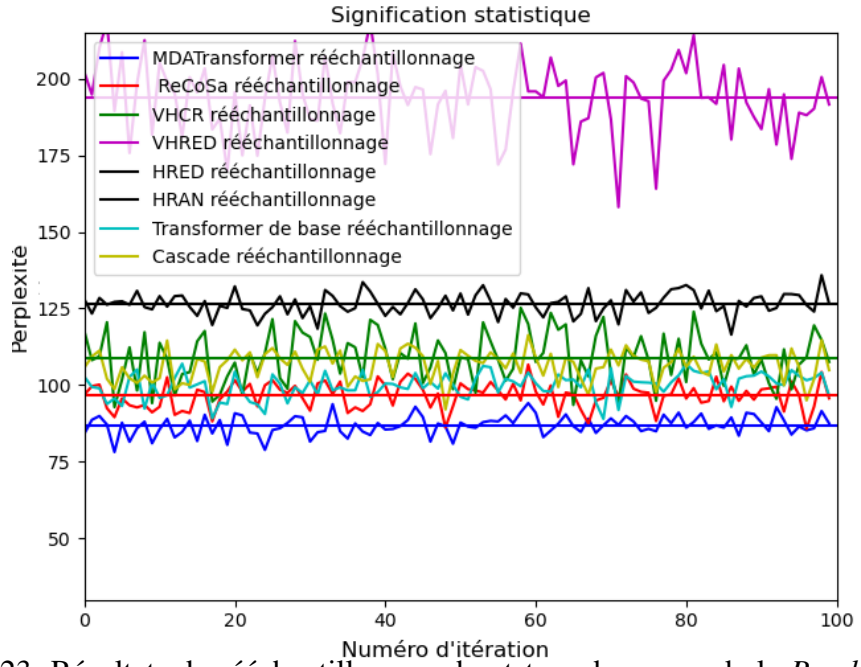


Figure 4.23: Résultats du rééchantillonnage bootstrap du scores de la *Perplexité* pour 100 ensembles de tests dérivés de corpus Ubuntu.

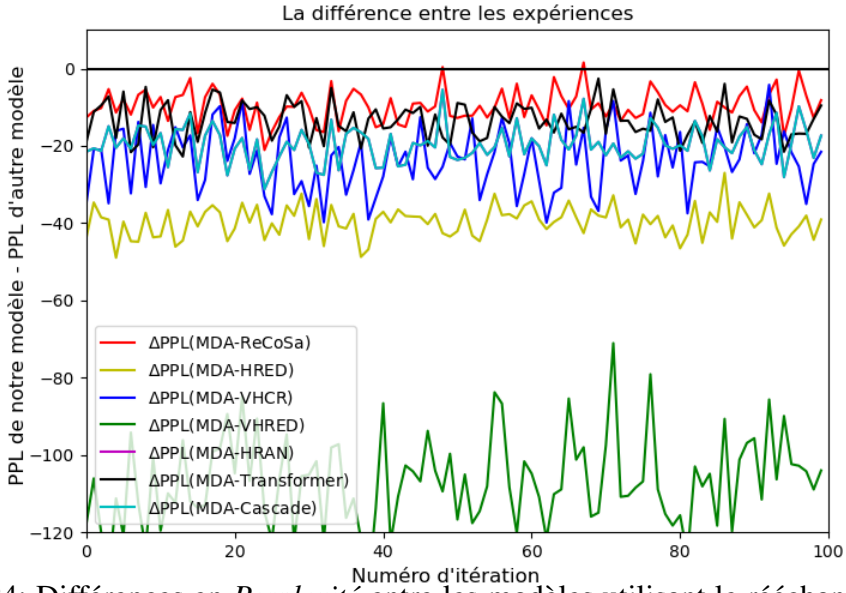


Figure 4.24: Différences en *Perplexité* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés de corpus Ubuntu.

tests dérivés du corpus Ubuntu et leurs valeurs moyennes pour chaque modèle. Les ré-

sultats montrent un intervalle de confiance à 95 % de [80.7, 92.9], [91.2, 106.1], [119.6, 132.6], [88.7, 106.6], [170.9, 213.4], [94.0, 123.5] et [87.8, 103.6] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa respectivement avec des valeurs moyennes de 87.8, 99.9, 126.4, 97.6, 193.6, 109.0 et 96.3. Ces valeurs sont très proches de celles rapportées dans le tableau 4.12 pour l'ensemble de tests Ubuntu : 87.41. De plus, la figure 4.24 montre une différence négative de *Perplexité* dans plus de 95% de rééchantillonnage utilisé entre notre modèle et les autres. Ceci confirme que notre modèle donne des valeurs de perplexité plus faibles.

Enfin, nous avons analysé la signification statistique du résultat en score *BLEU* puis la *Perplexité* de l'ensemble de test Cornell Movie.

La figure 4.25 fournit les graphes des scores *BLEU* obtenus en utilisant des tests dérivés du corpus Cornell Movie et leurs valeurs moyennes pour chaque modèle. Les résultats montrent un intervalle de confiance à 95 % de [1.2, 1.6], [0.1, 0.4], [0.3, 0.7], [1.0, 1.2], [0.4, 0.5], [0.5, 0.7] et [1.0, 1.3] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa respectivement avec des valeurs moyennes de 1.4, 0.7, 0.2 , 1.1, 0.5, 0.6 et 1.2. Ces valeurs sont très proches de celles rapportées dans le tableau 4.16 pour l'ensemble de tests Cornell Movie : 1.43. De plus, la figure 4.26 montre une différence de score *BLEU* constamment positive entre notre modèle et les autres.

La figure 4.27 fournit les graphes des valeurs de la *Perplexité* obtenus en utilisant de tests dérivés du corpus Cornell Movie et leurs valeurs moyennes pour chaque modèle. Les résultats montrent un intervalle de confiance à 95 % de [87.2, 100.6], [92.1, 106.2], [118.6, 131.0], [93.2, 109.3], [170.5, 203.0], [92.9, 120.5] et [92.5, 108.3] pour notre modèle, le Transformer de base, le HRED, le HRAN, le VHRED, le VHCR et le ReCoSa, avec des valeurs moyennes de 94.9, 125.2, 102.4, 104.3, 192.5, 107.0 et 103.3 respectivement. Ces valeurs sont très proches de celles rapportées dans le tableau 4.12

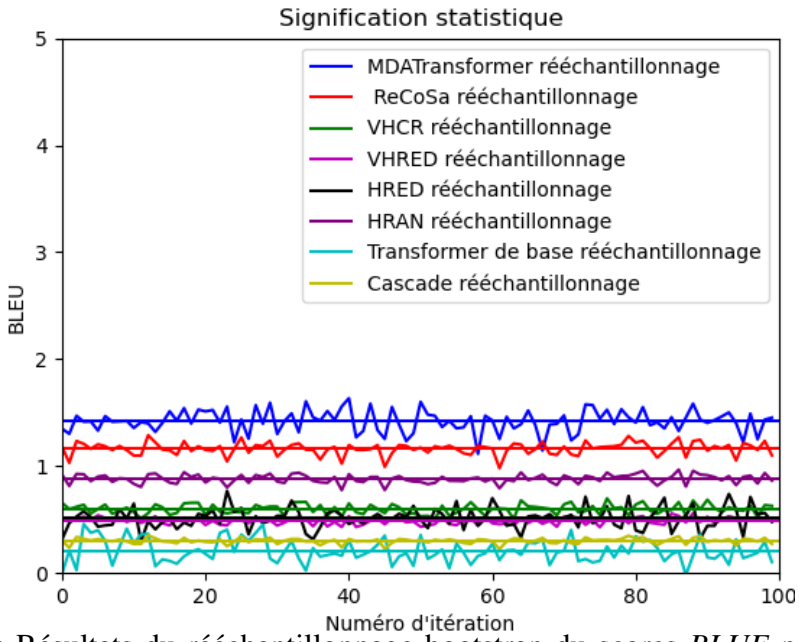


Figure 4.25: Résultats du rééchantillonnage bootstrap du scores *BLUE* pour 100 ensembles de tests dérivés du corpus Cornell Movie.

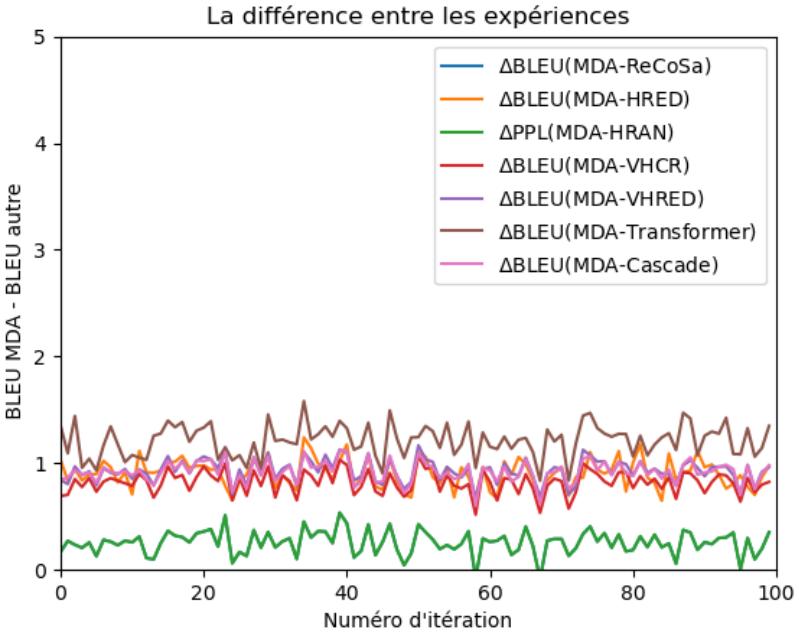


Figure 4.26: Différences en *BLUE* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés du corpus Cornell Movie.

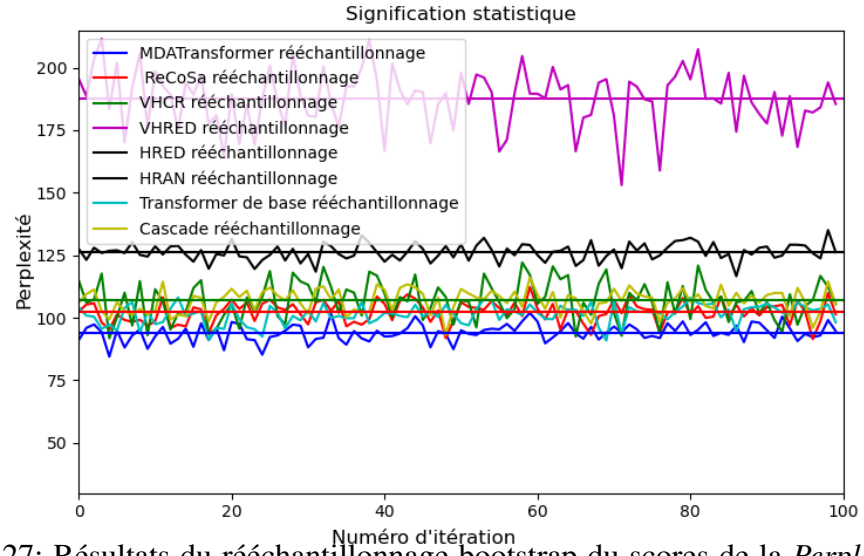


Figure 4.27: Résultats du rééchantillonnage bootstrap du scores de la *Perplexité* pour 100 ensembles de tests dérivés du corpus Cornell Movie .

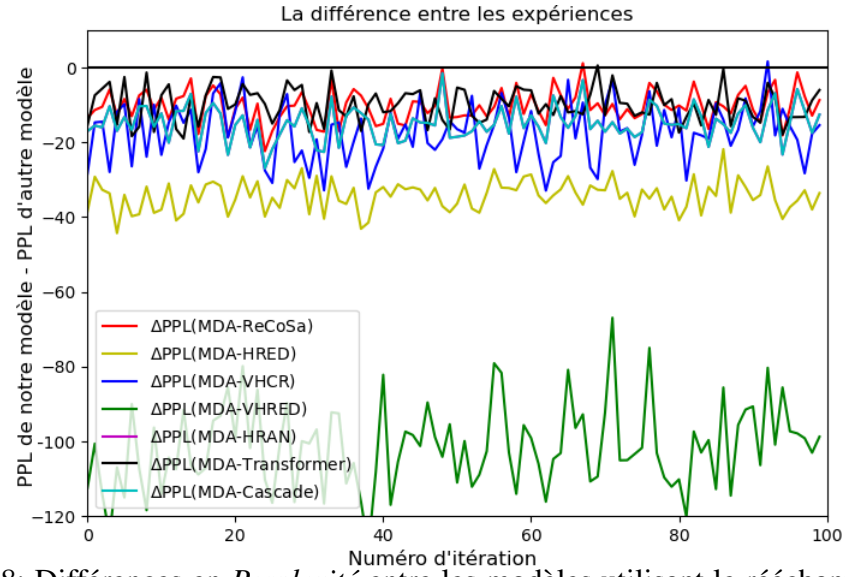


Figure 4.28: Différences en *Perplexité* entre les modèles utilisant le rééchantillonnage bootstrap pour 100 ensembles de tests dérivés du corpus Cornell Movie.

pour l'ensemble de tests Cornell Movie : 95.57. De plus, la figure 4.28 montre une différence négative de *Perplexité* dans plus de 95% de rééchantillonnage utilisé entre notre modèle et les autres. Ceci confirme que notre modèle donne des valeurs de perplexité plus faibles.

Tous les résultats des échantillonnages de toutes les figures montrent que la moyenne de la perplexité de notre modèle est inférieure à celles de tous les autres modèles et la moyenne du score BLEU est supérieure pour les trois ensembles de tests.

4.7 Évaluations sur la traduction automatique

La traduction automatique est considérée généralement comme étant une tâche plus simple que la génération de réponses dans les systèmes de conversation, car elle agit généralement sur une seule phrase d'entrée. Ainsi, un système capable de générer des dialogues devrait également être capable de gérer la traduction en langage naturel.

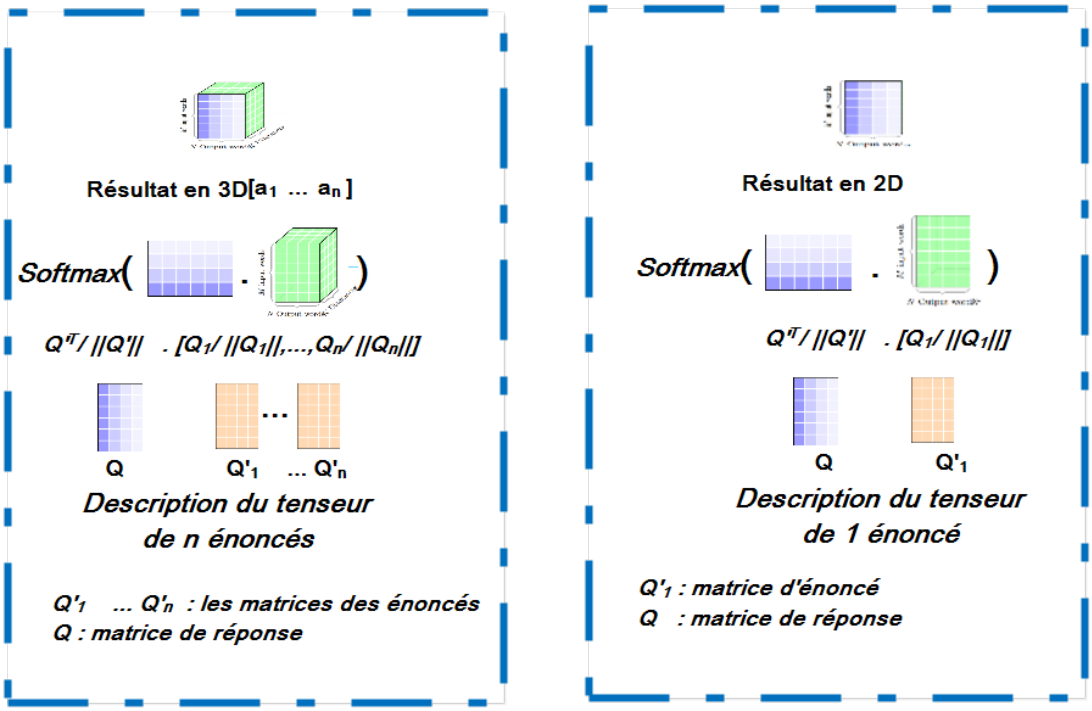


Figure 4.29: Différence entre l'attention du modèle qui utilise un seul énoncé comme phrase source pour la traduction (à droite) et l'attention du modèle qui utilise n énoncés (à gauche).

Dans notre modèle, étant donné que le modèle MDA proposé traite le dialogue d'entrée

par énoncé, la réduction de la profondeur du dialogue à un tour change le tenseur 3D illustré dans la figure 4.29 en une attention 2D du transformer d'origine, et il peut ensuite être utilisé dans les tâches de traduction comme le fait ce dernier.

Ce n'est pas le cas des autres modèles hiérarchiques où l'énoncé doit subir des changements dans la chaine d'encodeurs en raison de leurs traitements hiérarchiques avec plusieurs encodeurs, ce qui peut nuire au vecteur de contexte de la phrase source codée.

Nous avons étudié cela plus en détail avec les tâches de traduction automatique appliquée sur les paires de langues anglais-allemand et anglais-français, afin de les comparer avec le Transformer (Vaswani *et al.*, 2017). Nous avons conservé les mêmes paramètres et les corpus utilisés dans la section 4.3.1. Autrement dit, nous avons utilisé les tâches de traduction automatique *WMT 2014*[27] anglais-allemand et anglais-français (Bojar *et al.*, 2014). L'ensemble de données anglais-allemand [26] de 2014 comprenant environ 4,5 millions paires de phrases pour des vocabulaires source et cible partagent près de 37 000 mots.

L'ensemble de données anglais-français[26] de 2014 se composant d'environ 40 millions paires de phrases pour des vocabulaires source et cible partagent près de 32 000 mots (Zhou *et al.*, 2016). Nous avons utilisé le Parseur Moses pour diviser les différentes phrases [15].

Enfin, chacun des ensembles de développement[16] et de test[17] des deux tâches de traduction comprenait 3 003 phrases. Nous avons utilisé les mêmes hyperparamètres que ceux du document principal de Transformer de base (Vaswani *et al.*, 2017) comme la profondeur de séquence maximale à 120 pour toutes les phrases d'entrée et de sortie et pour tous les modèles. Le nombre des couches d'encodage et de décodage est égale à 6 pour le transformer MDA, à l'exception de la taille du lot qui a été réduite de 1024 à 70

26. https ://www.tensorflow.org/datasets/catalog/wmt14_translate

en raison de nos ressources de calcul limitées. La taille de plongement est égale à 512 appliquée pour tous les modèles.

Ceci nous permet de garantir que toute différence de performance entre les modèles serait uniquement en raison de la différence entre la représentation des données utilisées par les mécanismes d'attention multidimensionnelle et la représentation hiérarchique.

Deux expériences ont été appliquées sur le transformer MDA. La première utilise une profondeur de 1 énoncé, cela peut rendre le modèle similaire au modèle de transformer. La deuxième utilise une profondeur de 10 énoncés où le premier est la phrase source et les autres neuf énoncés sont remplis par le vide (on les remplit par le jeton <Pad>)

Le tableau 4.17 montre les résultats obtenus pour le score *BLEU* et la perplexité en utilisant les ensembles de test WMT-2014 [27] (*newtest2014* anglais-allemand et *newtest2014* anglais-français). Ces résultats confirment l'utilité de notre modèle pour les tâches de traduction, car c'était le seul proche du résultat retrouvé dans l'état de l'art en *BLEU* qui a été atteint par le transformer de base (Vaswani *et al.*, 2017).

Les scores *BLEU* (Papineni *et al.*, 2002) obtenus à partir de la traduction anglais-allemand, par les modèles hiérarchiques sont inférieurs à ceux obtenus par les modèles MDA (1 tour et 10 tours). Le meilleur résultat obtenu par ReCoSa est de 12,11. Ce score est presque 50% inférieur au résultat obtenu par les deux modèles proposés de MDA (1 tour), qui est de 24.26 et de MDA (10 tours), qui est de 23.41. Plus remarquable, le meilleur score *BLEU* obtenu pour la traduction de l'anglais vers le français des modèles hiérarchiques est celui de ReCoSa égale à 14.93. Celui-ci est aussi inférieur de 42% par rapport au modèle proposé de MDA (1 tour) qui est égale à 36.11 et par rapport au MDA (10 tours) qui est égale à 34.63.

Nous remarquons également que ReCoSa a obtenu un meilleur score *BLEU* pour la

27. http ://www.statmt.org/wmt14/test-filtered.tgz

	EN-GE		EN-FR	
	PPL	**BLEU**	**PPL**	**BLEU**
HRED	8.08	11.43	8.91	13.09
VHRED	8.19	10.68	9.26	12.54
VHCR	8.24	10.36	9.02	12.17
HRAN	7.93	11.94	8.43	13.82
ReCoSa	7.91	12.11	8.13	14.93
Cascade	7.92	12.03	7.96	14.76
MDA (1 Tour)	**6.23**	**24.26**	**6.48**	**36.11**
MDA (10 Tours)	6.29	23.41	6.71	34.63

Tableau 4.17: Perplexité et score *BLEU* des différents modèles pour la tâche de traduction EN-GE et EN-FR

traduction par rapport à tous les modèles hiérarchiques HRED, VHRED, VHCR et HRAN et même par rapport au modèle Cascade. ReCoSa a obtenu 12,11 pour la traduction anglais-allemand et 14,93 pour la traduction anglais-français. Ce modèle a réussi à surpasser Cascade qui a obtenu 12,03 et 14,76 pour les tâches de traduction anglais-allemand et anglais-français. Cela s'explique par l'utilisation de l'auto-attention dans l'architecture de ReCoSa.

Ces résultats en score *BLEU* obtenus à partir des modèles hiérarchiques montrent que ces derniers présentent quelques failles envers la traduction automatique. L'application des tâches de traduction automatique peut être une bonne méthode de comparaison entre les modèles de dialogue.

4.7.1 Analyse des erreurs

En examinant la traduction automatique de l'ensemble newtest2014 des modèles, plusieurs types d'erreurs ont été notées. Ceux-ci peuvent être résumées à travers les exemples du tableau 4.18. Les mots en gras dans le tableau représentent les mots générés par les

Exemple 1 EN-FR	**Langue source**	Frontier Airlines plans to charge up to $100 for passengers to store carry-on luggage on board their flight.
	Traduction	
	HRED	**<UNK> a l'intention** de **facturer** les passagers pour les bagages **<UNK>**
	VHRED	**les compagnies aériennes ont l'intention** de **facturer** les passagers **<UNK> <UNK>** .
	VHCR	**les compagnies aériennes prévoient** de facturer aux passagers leurs bagages à main . .
	HRAN	**les compagnies aériennes ont l'intention d'expédier** jusqu'à 100 $ de passagers .
	ReCoSa	**<UNK>** aériennes **prévoient** de **facturer** aux passagers jusqu'à **<UNK> <UNK>**
	Cascade	**les compagnies aériennes prévoient** de **facturer** aux passagers jusqu'à 100 $ **pour stocker** les bagages .
	MDA (1 Tour)	**<UNK>** airlines **prévoit** de **facturer** jusqu'à 100 $ aux passagers **pour stocker** leurs bagages à main **à bord** de **leur** vol .
	MDA (10 Tours)	**<UNK>** airlines **anticipe** de **facturer** jusqu'à 100 $ aux passagers **pour stocker** les bagages à main **à bord** de **leur** vol .
	Référence	Frontier Airlines envisage de faire payer jusqu'à 100 $ aux passagers qui transportent des bagages à main sur ses vols.
Exemple 2 EN-GE	**Langue source**	In Oregon, planners are experimenting with giving drivers different choices.
	Traduction	
	HRED	**verschiedene fahrmöglichkeiten** planer **im testschema** .(*divers planificateur d'options de conduite dans le schéma de test.*)
	VHRED	in **<UNK> testen** planer **verschiedene fahrmöglichkeiten** .(*à UNK, les planificateurs testent diverses options de conduite.*)
	VHCR	in **<UNK> testen** planer **verschiedene fahrmöglichkeiten** .(*à UNK, les planificateurs testent diverses options de conduite.*)
	HRAN	in **<UNK> testen** planer **verschiedene fahrmöglichkeiten** . (*à UNK, les planificateurs testent diverses options de conduite.*)
	ReCoSa	in **<UNK>** experimentieren planer damit, autofahrern eine reihe von **optionen anzubieten**.(*en UNK, les planificateurs expérimentent l'offre d'une gamme d'options de conduite.*)
	Cascade	in **<UNK>** experimentieren planer damit , autofahrern eine reihe von **optionen anzubieten**.(*en UNK, les planificateurs expérimentent l'offre d'une gamme d'options de conduite.*)
	MDA (1 Turn)	in **<UNK>** experimentieren planer damit , autofahrern eine reihe von **wahlmöglichkeiten** zu geben . (*Dans UNK, les planificateurs expérimentent en donnant à la conduite une gamme de choix.*)
	MDA (10 Turns)	in **<UNK>** experimentieren die planer damit , autofahrern eine reihe von **whlmöglichkeiten** zu geben. (*dans UNK, les planificateurs expérimentent en donnant à la conduite une gamme de choix.*)
	Référence	In Oregon experimentieren die Planer damit, Autofahrern eine Reihe von Auswahlmöglichkeiten zu geben.

Tableau 4.18: Exemples de traductions générées par les différents modèles pour les corpus de tests anglais-français et anglais-allemand

modèles et manquants de la référence.

La première section du tableau montre un exemple des traductions anglais-français. On peut y noter que les deux modèles de MDA, soit celui à 1 tour ou celui à 10 tours, ont remplacé le mot « *Frontier* » par l'étiquette <UNK>, puisque les mots hors vocabulaire ont été remplacés par cette étiquette. La seule différence entre les traductions générées par les deux modèles, est la traduction du mot *« plans »* en anglais par les mots en français *« prévoit »* généré par le premier modèle MDA (1 tour) et *« anticipe »* généré par le deuxième modèle MDA (10 tours). En effet, ces deux mots sont des synonymes du mot « *envisage* », utilisé dans la référence. En comparant ces traductions à celles générées par les modèles HRED, VHRED et HRAN, nous remarquons qu'elles sont plus courtes et incomplètes. Aussi, une partie de la phrase *« 100 $ aux passagers qui transportent des bagages à main sur ses vols. »*, est absente de la traduction. Le modèle HRAN a proposé une traduction incorrecte en introduisant le verbe « *expédier* » ; ce qui a provoqué un changement radical du sens de la phrase traduite. Ces exemples de traductions incorrectes, erronées ou incomplètes, contribuent dans la baisse du score *BLEU* lors des évaluations.

La deuxième section du tableau 4.18 montre un exemple des traductions anglais-allemand produites pour chaque modèle proposé MDA (1 tour) et MDA (10 tours). Nous avons ajouté des traductions, mises en italique, générées à l'aide de l'outil Google traduction à partir de l'allemand vers le français.

Nous remarquons dans cet exemple que les deux modèles de MDA n'ont pas traduit le mot « *Oregon* », comme une entité nommée de type location, qui représente le nom d'un État aux États-Unis. Les mots absents du corpus ou les mots rares ont été traités comme des mots hors du vocabulaire <UNK>. Un autre problème signalé est l'utilisation des synonymes des mots proposés dans la référence, tel que le mot « *wahlmöglichkeiten* », proposé par la traduction du MDA (1 tour et 10 tours) au lieu de « *Auswahlmöglichkei-*

	EN-FR	EN-DE
HRED	18.63	16.82
VHRED	17.41	14.91
VHCR	17.63	14.38
HRAN	19.02	16.78
ReCoSa	18.97	16.32
Cascade	19.14	16.81
MDA (1 tour)	**23.51**	**17.28**
MDA (10 tours)	23.38	17.12

Tableau 4.19: Nombre moyen de mots générés à partir des ensembles de test pour chaque modèle.

ten » utilisé dans la référence. Ces deux mots peuvent être utilisés comme traductions possibles du mot « *choices* ». Par contre, les modèles ReCoSa et Cascade proposent le mot « *optionen* » comme traduction pour le mot « *choices* » plutôt que « *Auswahlmöglichkeiten* » existant dans la référence. De plus, tous les modèles hiérarchiques proposent des traductions courtes avec moins de mots du vocabulaire partagés avec la référence. Ce type de problème affecte le résultat du score *BLEU* de notre modèle ainsi que celui des modèles hiérarchiques.

Nous notons que plusieurs concours (shared tasks) pour la tâche de traduction automatique, exigent plusieurs références (de 3 à 5) faites par des humains pour chaque phrase et d'une façon différente, afin d'établir un ensemble de test plus juste et pertinent pour l'utilisation de la métrique *BLEU* (Barrault *et al.*, 2019a). Malheureusement, dans notre cas, nous avions juste une référence pour chaque phrase dans nos ensembles test pour les deux paires de langues ; ce qui est pénalisant lors de la génération des termes de traductions synonymes à ceux de la référence.

Le tableau 4.19 montre le nombre moyen de mots générés à partir les corpus de tests anglais-français et anglais-allemand et pour chaque modèle. On peut remarquer que les modèles de MDA génère des réponses plus longues par rapport aux autres modèles.

Cependant, le modèle Cascade génère de longues phrases pour les corpus de test des deux taches, mais ne dépasse pas la moyenne des longueurs des phrases générées par les modèles MDA.

La différence de tailles entre les modèles hiérarchiques et les transformers MDA peut s'expliquer par le fait que les modèles hiérarchiques résument chaque énoncé dans un vecteur statique de taille fixe égale à $d_{modèle}$ (Serban *et al.*, 2016b, 2017c; Zhao *et al.*, 2017; Xing *et al.*, 2018; Zhang *et al.*, 2019a). Ce vecteur est créé par la dernière sortie du premier encodeur RNN. Ce vecteur n'apprend aucune information sur la longueur de la phrase source ou une association à la longueur de la phrase cible. En d'autres termes, une phrase de 10 mots est codée avec la même taille d'un vecteur du phrase à un seul mot. La figure 4.30 montre une architecture HRED utilisant un premier encodeur pour encoder deux phrases dans deux vecteurs de taille fixe h_1 et h_2 (Serban *et al.*, 2016b).

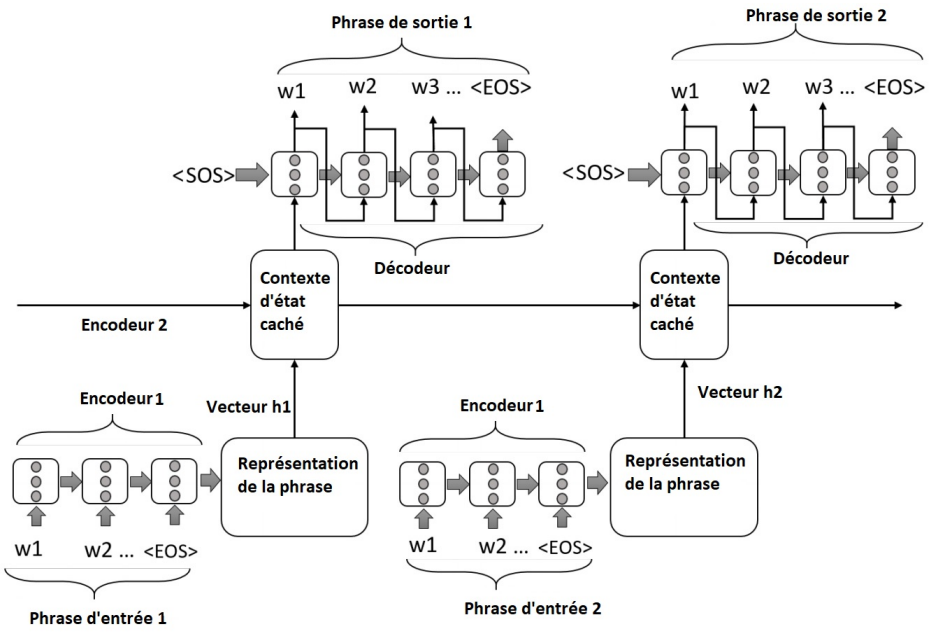


Figure 4.30: Graphe computationnel de l'architecture HRED pour une conversation de deux énoncés (Serban *et al.*, 2016b)

Cependant, des modèles de MDA génèrent des vecteurs dynamiques tels que chaque mot source est représenté par un vecteur de contexte dans l'encodeur.

Ensuite, le décodeur apprendra une correspondance directe entre les vecteurs des mots sources et cibles qui peuvent être représentés par la matrice d'attention. Dans cette attention chaque vecteur de mot dans l'ensemble source est associé à une ou plusieurs images de mots dans l'ensemble cible. Cela peut aider le décodeur à apprendre une association entre les longueurs de phrases sources et cibles.

L'association directe par une attention du MDA entre les vecteurs de mots d'une phrase source créés par l'encodeur et les vecteurs de mots d'une phrase cible générés par le décodeur explique l'augmentation des scores BLEU. En d'autres termes, le décodeur MDA apprend pour chaque mot source ses images cibles qui représentent le n-gramme ce qui augmente le score BLEU.

Tandis qu'un encodeur dans l'architecture hiérarchique qui encode une phrase source par un vecteur de taille fixe nous fait perdre des informations importantes sur la longueur de la phrase ainsi que sa composition linguistique. Le décodeur source il n'est pas obligé d'apprendre une image cible pour chaque mot source. Ceci est dû à l'absence de l'attention entre le décodeur et le premier encodeur, comme nous montre la figure 4.30 pour le HRED (Serban *et al.*, 2016b). Les autres architectures présentent le même problème (Serban *et al.*, 2016b, 2017c; Zhao *et al.*, 2017; Xing *et al.*, 2018; Zhang *et al.*, 2019a).

4.8 Discussion

Le tableau 4.20 montre les résultats des scores *BLEU* obtenus par les modèles « séquence à séquence » par les auteurs correspondants. On peut remarquer que les premiers modèles basés sur les réseaux récurrents, tels que décrits par Luong *et al.* (2015a), ont obtenu un score *BLEU* de 20,9 pour la tâche de traduction automatique WMT2014 et

	Score BLEU	**Auteurs**	**Modèle**
EN-FR	36.2	Bahdanau *et al.* (2015)	Séquence à Séquence qui utilise l'attention générale et une mémoire à long court terme (LSTM) multicouche
	35.6	Luong *et al.* (2015b)	Séquence à Séquence qui utilise l'attention linéaire et une mémoire à long et court terme (LSTM) multicouche
	36.5	Sutskever *et al.* (2014)	Séquence à Séquence utilise une mémoire à long et court terme (LSTM) multicouche
	37.7	Zhou *et al.* (2016)	Séquence à Séquence avec DeepAtt(modèles récurrents profonds avec connexions rapides pour la traduction), utilise une mémoire à long et court terme (LSTM) multicouche
	40.6	Gehring *et al.* (2017)	Séquence à Séquence avec convolution : utilise des blocs convolutifs et le mécanisme attention.
	38.1	Vaswani *et al.* (2017)	Transformer de base
	42.6	Tay *et al.* (2021)	Représentation omnidirectionnelle de l'attention pour le Transformer (OmniNet)
EN-GE	**Score BLEU**	**Auteurs**	
	20.9	Luong *et al.* (2015a)	Séquence à Séquence qui utilise l'attention linéaire et une mémoire à long et court terme (LSTM) multicouche
	20.7	Zhou *et al.* (2016)	Séquence à Séquence avec DeepAtt(modèles récurrents profonds avec connexions rapides pour la traduction), utilise une mémoire à long et court terme (LSTM) multicouche
	25.16	Gehring *et al.* (2017)	Séquence à Séquence avec convolution : utilise des blocs convolutifs et le mécanisme attention.
	28.3	Vaswani *et al.* (2017)	Transformer de base
	27.6	Kitaev *et al.* (2020)	Reformer
	29.8	Tay *et al.* (2021)	Représentation omnidirectionnelle de l'attention pour le Transformer (OmniNet)

Tableau 4.20: Scores *BLEU* obtenus dans l'état de l'art en utilisant les corpus WMT-2014 anglais-allemand et anglais-français.

la paire de langues EN-GE. De plus, le modèle simple « séquence à séquence » de Sutskever *et al.* (2014), a obtenu un score *BLEU* de 36.5 pour la paire de langues EN-FR et pour le corpus WMT2014. Ces scores sont presque supérieurs au double des

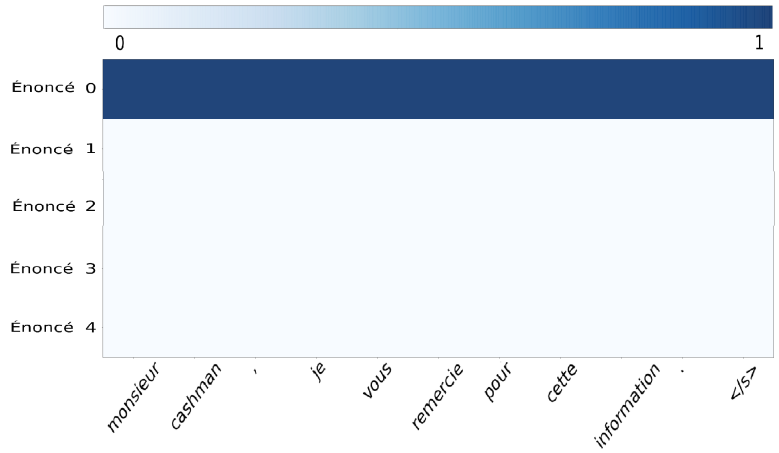


Figure 4.31: Visualisation de l'évolution de l'attention au niveau des cinq premiers énoncés pour un exemple de test de la traduction EN-FR avec un modèle de 10 énoncés. La visualisation est basée sur les cinq premiers énoncés, la première contient la phrase de la langue source ('*Mr Cashman, i am grateful for that information.*') et le reste est rempli avec des énoncés vides. la phrase de la langue cible est '*monsieur Cashman, je vous remercie pour cette information . </s>*

résultats obtenus par les modèles hiérarchiques. Ceci explique bien que les modèles hiérarchiques souffrent avec les tâches simples telles que la traduction automatique.

Cependant, notre modèle appliqué sur un énoncé de taille égale à 1 tour a contourné ce problème avec l'utilisation d'une attention adaptable pour la tâche de traduction automatique. Cette attention se transforme d'une attention 3D en une attention 2D avec l'écrasement de la taille de la dimension des énoncés à 1. L'utilisation d'une expérience de 10 tours (9 tours blancs et une phrase de la langue source) a eu un résultat proche de celui de 1 tour. Comme le montre les figures 4.32 et 4.31, l'attention dépend de la première phrase source qui contribue à la génération de mots pour la phrase cible. Généralement les autres phrases qui contiennent les mots vides (<blank>) ont la même valeur d'intensité de couleurs qui ne nous montre aucune contribution à la génération de mots. Par conséquence on peut conclure que le reste des phrases vides (<blank>)

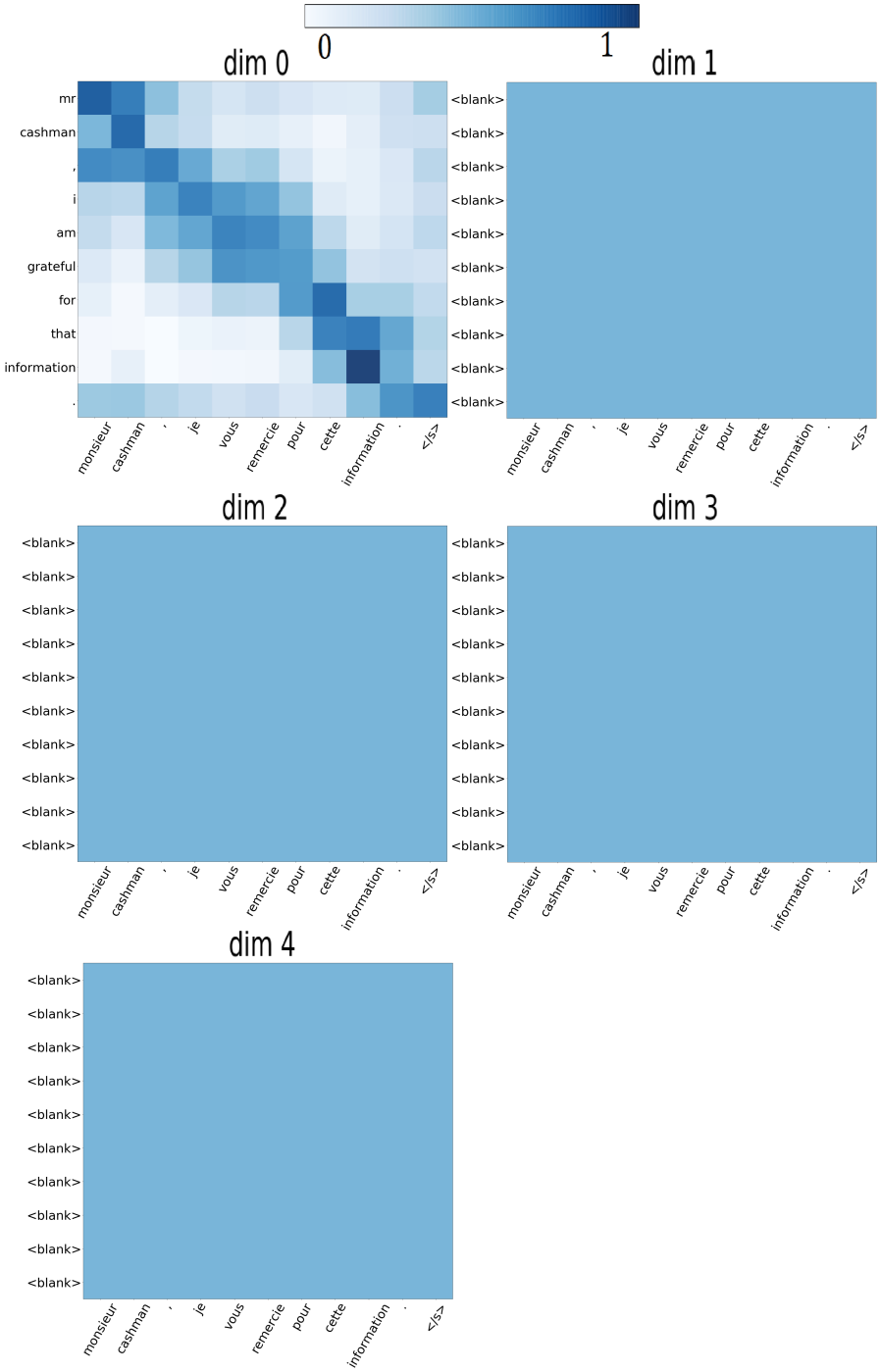


Figure 4.32: Visualisation de l'évolution détaillée de l'attention multidimensionnelle pour un exemple de test de la traduction EN-FR avec un modèle de 10 énoncés. La visualisation est basée sur les cinq premiers énoncés, la première contient la phrase de la langue source ('*Mr Cashman, i am grateful for that information*') et le reste est rempli avec des énoncés vides. La phrase de la langue cible est '*monsieur Cashman, je vous remercie pour cette information . <s>*.

sont ignorées.

De plus, nous avons constaté que le modèle n'a pas réussi à trouver un résultat égal à celui du transformer que nous avons obtenu dans la section 4.3.1 de 27.28 pour la paire de langues anglais-allemand et 38.06 pour la paire de langues anglais-français. Cela est peut-être dû à la légère différence entre les deux modèles qui s'explique par l'ajout au transformer MDA, d'un réseau linéaire qui transforme des vecteurs de contexte 3D en 2D. Une autre légère différence est également constatée dans les deux codages positionnels utilisés pour ce modèle.

Avec les résultats améliorés en terme du *BLEU* score, pour la tâche de traduction automatique impliquant les deux paires de langues, notre modèle proposé répond au critère selon lequel un bon modèle de dialogue peut très bien être adapté à la tâche de traduction automatique.

De plus, notre modèle est parallélisable en séquence sauf durant l'étape de l'encodage où un lot des énoncés doit passer un à la fois dans l'encodeur. Cela est dû à l'élimination des modules RNRs de la construction architecturale avec tous ses inconvénients.

Un autre avantage important du modèle que nous proposons, est celui de sa capacité à déduire les sources des mots générés permettant à l'être humain de retracer à l'oeil les relations entre les mots. Cela permet de trouver les relations causales entre les mots sources et cibles.

CHAPITRE V

CONCLUSION

Les travaux décrits dans cette thèse impliquent l'utilisation des techniques les plus récentes du domaine de l'intelligence artificielle, dont celles issues de l'apprentissage profond et du traitement automatique du langage naturel ; et ce afin de concevoir et de développer des architectures neuronales et algorithmiques innovatrices et efficaces, en liaison avec les agents conversationnels génératifs multi-tours.

Dans cette section, nous résumons les principales contributions (section 5.1), les contraintes et les obstacles rencontrés lors de la conduite de ces travaux de recherche (section 5.2) ainsi que les perspectives (section 5.3).

5.1 Contributions

Dans cette thèse, nous avons proposé trois idées originales afin d'améliorer l'architecture du Transformer avec des applications sur les deux tâches de génération de réponses et de traduction automatique neuronale.

En premier lieu, nous avons proposé *Cascade*, un nouveau modèle génératif de réponses en langage naturel, incorporé dans les systèmes de dialogue à domaine ouvert. Nous avons démontré, à travers diverses évaluations, son efficacité globale tout en le comparant aux autres modèles de l'état-de-l'art, à la pointe de la technologie, avec une application sur une variété de corpus d'apprentissage. Aussi, notre proposition originale

d'une *attention multidimensionnelle*, a aussi aidé dans l'amélioration des performances de prédiction dans les systèmes de conversation multi-tours, comme démontré par nos évaluations. Néanmoins, le modèle proposé a montré des limites sur le nombre des énoncés utilisés durant l'encodage. Aussi, sa très grande complexité par rapport aux autres modèles constitue un défi majeur lors de sa conception.

En revanche, ces dernières années, nous avons observé l'émergence de modèles efficaces, remplaçant les RNR par un mécanisme d'auto-attention, proposés pour la tâche de traduction automatique. Afin de profiter davantage de ces modèles avec mécanisme d'auto-attention, nous avons commencé par une optimisation du système d'attention en proposant un modèle réduit du Transformer. D'un côté, cette amélioration est basée sur la correspondance directe des espaces de représentation d'entrée et de sortie. D'un autre côté, elle s'appuie sur une attention tenant compte de la similarité cosinus remplaçant les espaces de représentation de requête-clé-valeur trouvée dans d'autres architectures des Transformers, y compris le récent modèle Reformer.

Notre approche proposée présente divers avantages, dont l'économie sur la matrice de projection et l'élimination du besoin d'un facteur de mise à l'échelle heuristique, par l'utilisation de la similitude cosinus. Cela mène à des performances de traduction de séquence améliorées par rapport aux modèles originaux, comme démontré dans les évaluations sur la tâche de traduction automatique neuronale pour les deux paires de langues anglais-allemand et anglais-français.

En deuxième lieu, nous avons proposé *une architecture étendue du Transformer avec une attention multidimensionnelle*. Ce nouveau modèle génératif est destiné à la génération de réponses en langage naturel dans les systèmes de dialogue de bout en bout. Nous avons montré l'efficacité globale de ce modèle à travers des évaluations en le comparant aux modèles de l'état-de-l'art avec une application sur une variété de corpus d'apprentissage. De plus, le mécanisme d'attention multidimensionnelle proposé

a amélioré non seulement les performances de prédiction des systèmes de conversation multi-tours comme démontré dans nos évaluations ; mais il a offert également la possibilité d'aller plus loin en perçant la complexité du modèle de prédiction neuronale, tout en identifiant toutes les sources des mots générés. Les comparaisons faites avec les modèles de l'état-de-l'art favorisent l'hypothèse selon laquelle un encodage direct des informations d'entrée est préférable à l'utilisation d'une chaîne d'encodeurs dans une forme hiérarchique. Ces comparaisons confirment à nouveau la supériorité du Transformer sur les architectures encodeur-décodeur basée sur les RNRs, en termes de complexité du calcul et de précision de sortie.

Ainsi, les principales contributions qui résultent des propositions ci-dessus sont les suivantes :

1. Une architecture encodeur-décodeur qui utilise un seul encodeur et un seul décodeur afin de minimiser les étapes de la chaîne d'encodage des énoncés et ainsi éliminer le potentiel de propagation et d'amplification des erreurs entre les niveaux d'encodage ;
2. Un mécanisme d'ordonnancement des plongements de mots d'entrée par un double encodage explicite de position ; l'un pour leurs positions dans les énoncés respectifs et l'autre pour les positions des énoncés dans le dialogue ;
3. Un mécanisme de traçage du contexte utilisant l'attention multidimensionnelle. Ce mécanisme peut refléter les alignements de causalité entre les mots sources et les mots cibles sous forme de cartes de chaleur. En d'autres termes, des causalités sémantiques sont générées au niveau du dialogue au complet, permettant ainsi de retracer des mots sources derrière chaque sortie générée à la fois, avec leurs positions dans les énoncés et leurs provenances, pour une meilleure prédiction et une meilleure interprétabilité du modèle ;
4. Une architecture neuronale qui permet autant la génération de réponses dans les systèmes de dialogues multi-tours, que la traduction automatique neuronale de

séquences de phrases.

5.2 Contraintes et Obstacles rencontrés

Le premier obstacle rencontré dans la réalisation de notre projet de thèse reste l'évaluation manuelle utilisée pour comparer les agents conversationnels. L'évaluation manuelle est non reproductible et coûteuse en termes de temps et de ressources financières.

De plus, le grand nombre d'énoncés comparés (100 énoncés générés pour 7 modèles en utilisant les trois corpus) devient une tâche fastidieuse pour les évaluateurs. Le temps de comparaison a nécessité plus d'un mois de travail pour chaque évaluateur ; contrairement aux évaluations automatiques qui nécessitent juste le temps de développement et d'exécution.

Le deuxième obstacle est la disponibilité des ressources computationnelles. Les modèles utilisés consomment des ressources GPU très coûteuses. De plus, ces modèles ne sont pas totalement parallélisables ; et les modèles hiérarchiques prennent énormément de temps même avec l'utilisation de GPU. Ceci est dû à l'architecture récurrente des modèles où une génération d'un mot nécessite la génération de tous les mots précédents notamment avec de gros corpus comme celui de la traduction.

Le troisième obstacle est la disponibilité des corpus d'apprentissage et la comparaison aux travaux les plus récents de l'état de l'art. Comme le domaine de l'intelligence artificielle est notamment lié à d'importantes avancées technologiques qui ont permis d'accroître de façon considérable les avancées dans les domaines de l'apprentissage automatique et du traitement automatique du langage naturel ; comme les deux tâches de la traduction automatique neuronale et de la génération des réponses sont considérées comme des tâches très convoitées dans la recherche et très liées par les avancements de l'apprentissage profond ; nous étions confrontées pendant ces quatre années d'études et de recherche, au problème de comparaisons avec les modèle d'apprentissage pro-

fond les plus récents à chaque fois. Ceci n'a pas été facile comme nous étions liés au problème des ressources computationnelles afin d'implémenter les systèmes et des évaluations manuelles.

5.3 Perspectives

Dans les travaux futurs, nous visons l'amélioration du modèle proposé avec des techniques d'optimisation telles qu'utilisées dans le Reformer. Aussi, nous visons l'étude des langues très peu dotées, comme les langues autochtones et en danger, ainsi que d'autres applications du traitement automatique du langage naturel, tels que la compréhension du langage naturel et les résumés automatiques. En outre, l'étude des nouveaux algorithmes exploitant les modèles *BERT* (Bidirectional Encoder Representations from Transformers) et *GPT-3* (Generative Pre-trained Transformer 3) pour la traduction automatique neuronale, représentee une autre direction à explorer.

Aussi, nous visons à adapter le modèle MDA aux problèmes multitâches avec plus d'un utilisateur dans plus d'une langue, telle que les différents dialectes écrits par les utilisateurs, ou des textes multilingues souvent utilisés dans les réseaux sociaux.

En dernier, nous visons l'incorporation des émotions dans les systèmes de génération de réponses afin de concevoir un agent conversationnel émotionnel qui aiderait à générer des réponses plus améliorées en tenant compte des émotions et de l'intention de l'usager.

Annexe

PUBLICATIONS

Liste des publications scientifiques qui rentrent dans le cadre des travaux de recherche de cette thèse :

Billal Belainine, Fatiha Sadat, Mounir Boukadoum. (2022). Multi-Dimension Attention for Multi-Turn Dialog Generation, *In Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 11. 2022 (Student abstract).*

Billal Belainine, Fatiha Sadat, Mounir Boukadoum. (2022). End-to-End Dialog Generation Using a Single Encoder and a Decoder Cascade with a Multi-Dimension Attention Mechanism *In IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS), doi : 10.1109/TNNLS.2022.3151347.*

Billal Belainine, Fatiha Sadat, Mounir Boukadoum, Hakim Lounis. (2020). Towards a Multi-Dataset for Complex Emotions Learning based on Deep Neural Networks. *Proceedings of the Second Workshop on Linguistic and Neurocognitive Resources. 2020.*

Billal Belainine, Fatiha Sadat, Hakim Lounis. (2020). Modelling a Conversational Agent with Complex Emotional Intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 10. 2020. Doctoral Consortium Track.*

Billal Belainine, Fatiha Sadat, Hakim Lounis. (2020). Complex Emotional Intelligence Learning Using Deep Neural Networks (Student Abstract). *Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 10. 2020.*

Billal Bilainine, Fonseca Alexandro, Fonseca Alexandro, , Fatiha Sadat. (2016). Named entity recognition and hashtag decomposition to improve the classification of tweets. *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT), 102-111 Dec 2016.*

RÉFÉRENCES

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C. et Parikh, D. (2015). Vqa : Visual question answering. Dans *The IEEE International Conference on Computer Vision (ICCV)*.

Asri, L. E., Schulz, H., Sharma, S., Zumer, J., Harris, J., Fine, E. et Suleman, K. (2017). Frames : A corpus for adding memory to goal-oriented dialogue systems. Dans *The Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Ba, J. L., Kiros, J. R. et Hinton, G. E. (2016). Layer normalization. Dans *Conference on Neural Information Processing Systems (NeurIPS)*.

Bahdanau, D., Cho, K. et Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Dans Y. Bengio et Y. LeCun (dir.). *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Banchs, R. E. (2012). Movie-DiC : a movie dialogue corpus for research and development. Dans *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 203–207., Jeju Island, Korea. Association for Computational Linguistics.

Barrault, L., Bojar, O., Costa-Jussa, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S. *et al.* (2019a). Findings of the 2019 conference on machine translation (wmt19). Dans *Proceedings of the Fourth Conference on Machine Translation (Volume 2 : Shared Task Papers, Day 1)*, 1–61.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M. et Zampieri, M. (2019b). Findings of the 2019 conference on machine translation (WMT19). Dans *Proceedings of the Fourth Conference on Machine Translation (Volume 2 : Shared Task Papers, Day 1)*, 1–61., Florence, Italy. Association for Computational Linguistics.

Belainine, B., Sadat, F. et Boukadoum, M. (2022a). End-to-end dialogue generation using a single encoder and a decoder cascade with a multidimension attention mechanism. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11. http://dx.doi.org/10.1109/TNNLS.2022.3151347

Belainine, B., Sadat, F. et Boukadoum, M. (2022b). Multi-dimension attention for multi-turn dialog generation (student abstract). Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 12909–12910.

Belainine, B., Sadat, F., Boukadoum, M. et Lounis, H. (2020a). Towards a multi-dataset for complex emotions learning based on deep neural networks. Dans *Proceedings of the Second Workshop on Linguistic and Neurocognitive Resources*, 50–58., Marseille, France. European Language Resources Association.

Belainine, B., Sadat, F. et Lounis, H. (2020b). Complex emotional intelligence learning using deep neural networks (student abstract). Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 13755–13756.

Belainine, B., Sadat, F. et Lounis, H. (2020c). Modelling a conversational agent with complex emotional intelligence. Dans *AAAI-20*. doctoral consortium poster sessions, 2 pages.

Beltagy, I., Peters, M. E. et Cohan, A. (2021). Longformer : The long-document transformer. *arXiv :2004.05150*.

Bengio, Y., Ducharme, R., Vincent, P. et Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H. et Winograd, T. (1977). Gus, a frame-driven dialog system. *Artificial intelligence*, *8*(2), 155–173.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L. et Tamchyna, A. s. (2014). Findings of the 2014 workshop on statistical machine translation. Dans *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 12–58., Baltimore, Maryland, USA. Association for Computational Linguistics.

Bonetta, G., Cancelliere, R., Liu, D. et Vozila, P. (2021). Retrieval-augmented transformer-xl for close-domain dialog generation. *The International FLAIRS Conference Proceedings*, *34*(1).

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R. et Bengio, S. (2016). Generating sentences from a continuous space. Dans *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Bradeško, L. et Mladenić, D. (2012). A survey of chatbot systems through a loebner prize competition. Dans *Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies*, 34–37.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. et Amodei, D. (2020). Language models are few-shot learners. Dans H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, et H. Lin (dir.). *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Bruni, E. et Fernandez, R. (2017). Adversarial evaluation for open-domain dialogue generation. Dans *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 284–288.

Cahn, J. (2017). *CHATBOT : Architecture, Design, & Development*. (Thèse de doctorat). University of Pennsylvania.

Callison-Burch, C., Koehn, P., Monz, C. et Zaidan, O. (2011). Findings of the 2011 workshop on statistical machine translation. Dans *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 22–64.

Carpenter, R. . (2017). Cleverbot. Récupéré de `http://www.cleverbot.com`

Carreto Fidalgo, D., Vila-Suero, D., Aranda Montes, F. et Talavera Cepeda, I. (2021). System description for ProfNER - SMMH : Optimized finetuning of a pretrained transformer and word vectors. Dans *Proceedings of the Sixth Social Media Mining for Health (#SMM4H) Workshop and Shared Task*, 69–73., Mexico City, Mexico. Association for Computational Linguistics.

Chakrabarti, A., Satuluri, V., Srivathsan, A. et Parthasarathy, S. (2015). A bayesian perspective on locality sensitive hashing with extensions for kernel methods. *ACM Trans. Knowl. Discov. Data*, *10*(2).

Cheng, J., Dong, L. et Lapata, M. (2016). Long short-term memory-networks for machine reading. Dans *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 551–561., Austin, Texas. Association for Computational Linguistics.

Chung, J., Gulcehre, C., Cho, K. et Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. Dans *NIPS 2014 Workshop on Deep Learning, December 2014*.

Colby, K. M., Hilf, F. D., Weber, S. et Kraemer, H. C. (1972). Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, *3*, 199–221.

Correia, G. M., Niculae, V. et Martins, A. F. T. (2019). Adaptively sparse transformers. Dans *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 2174–2184.

Cuayáhuitl, H., Lee, D., Ryu, S., Cho, Y., Choi, S., Indurthi, S., Yu, S., Choi, H., Hwang, I. et Kim, J. (2019). Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing*, *366*, 118–130.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V. et Salakhutdinov, R. (2019). Transformer-xl : Attentive language models beyond a fixed-length context. Dans *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1 : Long Papers*, 2978–2988.

Danescu-Niculescu-Mizil, C. et Lee, L. (2011). Chameleons in imagined conversations : A new approach to understanding coordination of linguistic style in dialogs. Dans *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. et Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391–407.

Denkowski, M. et Lavie, A. (2011). Meteor 1.3 : Automatic metric for reliable optimization and evaluation of machine translation systems. Dans *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 85–91., Edinburgh, Scotland. Association for Computational Linguistics.

Devlin, J. et Chang, M.-W. (2018). Open sourcing bert : State-of-the-art pre-training for natural language processing. *Google AI Blog. Weblog.[Online] Available from : https ://ai. googleblog. com/2018/11/open-sourcing-bertstate-of-art-pre. html [Accessed 4 December 2019]*.

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. Dans *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186., Minneapolis, Minnesota. Association for Computational Linguistics.

Dušek, O. (2017). *Novel Methods for Natural Language Generation*. (Thèse de doctorat). Faculty of Mathematics and Physics, Charles University, Prague.

Erkan, G. et Radev, D. R. (2004). Lexrank : Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, *22*, 457–479.

Fan, H., Wang, J., Zhuang, B., Wang, S. et Xiao, J. (2019). A hierarchical attention based seq2seq model for chinese lyrics generation. Dans A. C. Nayak et A. Sharma (dir.). *PRICAI 2019 : Trends in Artificial Intelligence*, 279–288., Cham. Springer International Publishing.

Fan, Y., Xie, S., Xia, Y., Wu, L., Qin, T., Li, X. et Liu, T. (2020). Multi-branch attentive transformer. Dans *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7839–7843.

Forchini, P. (2013). Using movie corpora to explore spoken american english. *Variation and Change in Spoken and Written Discourse : Perspectives from corpus linguistics*, *21*, 123.

Forgues, G., Pineau, J., Larchevêque, J.-M. et Tremblay, R. (2014). Bootstrapping dialog systems with word embeddings. Dans *Nips, modern machine learning and natural language processing workshop*, volume 2.

Fotsoh, A. (2018). *Recherche d'entités nommées complexes sur le Web - propositions pour l'extraction et pour le calcul de similarité*. (Thèse de doctorat).

Gao, J., Galley, M. et Li, L. (2018a). *Neural Approaches to Conversational AI*, Dans *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, (p. 1371–1374). Association for Computing Machinery : New York, NY, USA.

Gao, J., Galley, M. et Li, L. (2018b). *Neural Approaches to Conversational AI*, Dans *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, (p. 1371–1374). Association for Computing Machinery : New York, NY, USA.

Gehring, J., Auli, M., Grangier, D., Yarats, D. et Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. Dans D. Precup et Y. W. Teh (dir.). *Proceedings of the 34th International Conference on Machine Learning*, volume 70 de *Proceedings of Machine Learning Research*, 1243–1252. PMLR.

Gomez, A. N., Ren, M., Urtasun, R. et Grosse, R. B. (2017). The reversible residual network : Backpropagation without storing activations. Dans I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett (dir.). *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Graves, A. (2012). Sequence transduction with recurrent neural networks. Dans *The 29th International Conference on Machine Learning (ICML)*.

Gu, Y., Wen, J., Sun, H., Song, Y., Ke, P., Zheng, C., Zhang, Z., Yao, J., Zhu, X., Tang, J. et Huang, M. (2022). Eva2.0 : Investigating open-domain chinese dialogue systems with large-scale pre-training. *arXiv preprint arXiv :2108.01547*.

Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X. et Zhang, Z. (2019). Star-transformer. Dans *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

He, K., Zhang, X., Ren, S. et Sun, J. (2016). Deep residual learning for image recognition. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. et Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hinton, G., Rumelhart, D. et Williams, R. (1986). Learning internal representations by error propagation. *Parallel distributed processing*, *1*, 318–362.

Hoang, D.-T. et Kang, H.-J. (2019). Rolling element bearing fault diagnosis using convolutional neural network and vibration image. *Cognitive Systems Research*, *53*, 42–50.

Hochreiter, S. et Schmidhuber, J. (1997). Lstm can solve hard long time lag problems. Dans *Advances in neural information processing systems*, 473–479.

Honghao, W., Zhao, Y. et Ke, J. (2017). Building chatbot with emotions. Dans *Stanford*. Récupéré de `http://web.stanford.edu/class/cs224s/reports/Honghao_Wei.pdf`

Hu, W., Le, R., Liu, B., Ma, J., Zhao, D. et Yan, R. (2020). Translation vs. dialogue : A comparative analysis of sequence-to-sequence modeling. Dans *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Huang, C., Zaïane, O. R., Trabelsi, A. et Dziri, N. (2018). Automatic dialogue generation with expressed emotions. Dans *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, 49–54.

Huang, L., Chen, W. et Qu, H. (2021). *Accelerating Transformer for Neural Machine Translation*, Dans *2021 13th International Conference on Machine Learning and Computing*, (p. 191–197). Association for Computing Machinery : New York, NY, USA.

Huang, M., Zhu, X. et Gao, J. (2020). Challenges in building intelligent open-domain dialog systems. *ACM Trans. Inf. Syst.*, *38*(3).

Irie, K., Gerstenberger, A., Schlüter, R. et Ney, H. (2020). How much self-attention do we needf trading attention for feed-forward layers. Dans *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6154–6158.

Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R. et Daumé III, H. (2014). A neural network for factoid question answering over paragraphs. Dans *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 633–644.

Jafarpour, S. et Burges, C. J. (2010). *Filter, Rank, and Transfer the Knowledge : Learning to Chat*. Rapport technique MSR-TR-2010-93

Jiang, Z., Gao, S. et Chen, L. (2020). Study on text representation method based on deep learning and topic information. *Computing*, *102*, 623–642.

Joshi, N., Jingfei, G., Mandar, D., Lewis, D., Chen, O., Levy, M., Liu, Luke Zettlemoyer, V., Stoyanov, Y. et Ott, M. (2020). Roberta : A robustly optimized bert pretraining approach. Dans *International Conference on Learning Representations.*

Jozefowicz, R., Zaremba, W. et Sutskever, I. (2015). An empirical exploration of recurrent network architectures. Dans *International conference on machine learning*, 2342–2350.

Jurafsky, D. et Martin, J. (2017). *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2.

Kalchbrenner, N. et Blunsom, P. (2013). Recurrent continuous translation models. Dans *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709., Seattle, Washington, USA. Association for Computational Linguistics.

Kalchbrenner, N., Danihelka, I. et Graves, A. (2016). Grid long short-term memory. Dans *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Khusainova, A., Khan, A., Rivera, A. R. et Romanov, V. (2021). Hierarchical transformer for multilingual machine translation. Dans M. ZampieriOmnidirectional, P. Nakov, N. Ljubesic, J. Tiedemann, Y. Scherrer, et T. Jauhiainen (dir.). *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects, VarDial@EACL 2021, Kiyv, Ukraine, April 20, 2021*, 12–20. Association for Computational Linguistics.

Kitaev, N., Kaiser, L. et Levskaya, A. (2020). Reformer : The efficient transformer. Dans *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. Dans *Proceedings of the 2004 conference on empirical methods in natural language processing*, 388–395.

Koehn, P. (2005). Europarl : A parallel corpus for statistical machine translation. Dans *MT summit*, volume 5, 79–86. Citeseer.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R. *et al.* (2007). Moses : Open source toolkit for statistical machine translation. Dans *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, 177–180. Association for Computational Linguistics.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. et Soricut, R. (2020). ALBERT : A lite BERT for self-supervised learning of language representations. Dans *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Lavie, A. et Agarwal, A. (2007). Meteor : An automatic metric for mt evaluation with high levels of correlation with human judgments. Dans *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, 228–231., Stroudsburg, PA, USA. Association for Computational Linguistics.

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, M., Frank, L., Beute, F., de Kort, Y. et IJsselsteijn, W. (2017). Bots mind the social-technical gap. Dans *Proceedings of 15th European Conference on Computer-Supported Cooperative Work-Exploratory Papers*. European Society for Socially Embedded Technologies (EUSSET).

Leuski, A. et Traum, D. (2011). Npceditor : Creating virtual human dialogue using information retrieval techniques. *Ai Magazine*, *32*(2), 42–56.

Li, J., Galley, M., Brockett, C., Gao, J. et Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. Dans *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, San Diego California, USA, June 12-17, 2016*, 110–119.

Li, J., Galley, M., Brockett, C., Spithourakis, G. P., Gao, J. et Dolan, W. B. (2016b). A persona-based neural conversation model. Dans *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1 : Long Papers*.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M. et Gao, J. (2016c). Deep reinforcement learning for dialogue generation. Dans *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 1192–1202.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A. et Jurafsky, D. (2017a). Adversarial learning for neural dialogue generation. Dans *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 2157–2169.

Li, Y., Su, H., Shen, X., Li, W., Cao, Z. et Niu, S. (2017b). Dailydialog : A manually labelled multi-turn dialogue dataset. Dans *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Lin, C.-Y. (2004). ROUGE : A package for automatic evaluation of summaries. Dans *Text Summarization Branches Out*, 74–81., Barcelona, Spain. Association for Computational Linguistics.

Liu, C., Lowe, R., Serban, I., Noseworthy, M., Charlin, L. et Pineau, J. (2016). How NOT to evaluate your dialogue system : An empirical study of unsupervised evaluation metrics for dialogue response generation. Dans *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 2122–2132.

Liu, X., Yu, H.-F., Dhillon, I. et Hsieh, C.-J. (2020). Learning to encode position for transformer with continuous dynamical model. Dans H. D. III et A. Singh (dir.). *Proceedings of the 37th International Conference on Machine Learning*, volume 119 de *Proceedings of Machine Learning Research*, 6327–6335. PMLR.

Lowe, R. T., Pow, N., Serban, I. V., Charlin, L., Liu, C.-W. et Pineau, J. (2017). Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, *8*(1), 31–65.

Luong, T., Pham, H. et Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 1412–1421.

Luong, T., Sutskever, I., Le, Q. V., Vinyals, O. et Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. Dans *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1 : Long Papers*, 11–19.

McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McNeal, M. L. et Newyear, D. (2013). Chatbot creation options. *Library Technology Reports*, *49*(8), 11 – 17.

Meffert, K. (2006). Supporting design patterns with annotations. Dans *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on*, 8–pp. IEEE.

Mehri, S. et Eskénazi, M. (2020). USR : an unsupervised and reference free evaluation metric for dialog generation. Dans *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 681–707.

Meister, C., Vieira, T. et Cotterell, R. (2020). Best-First Beam Search. *Transactions of the Association for Computational Linguistics*, *8*, 795–809.

Mi, F., Li, Y., Zeng, Y., Zhou, J., Wang, Y., Xu, C., Shang, L., Jiang, X., Zhao, S. et Liu, Q. (2022). Pangubot : Efficient generative dialogue pre-training from pre-trained language model. *arXiv preprint arXiv :2203.17090*.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. et Khudanpur, S. (2010). Recurrent neural network based language model. Dans *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J. et Khudanpur, S. (2011). Extensions of recurrent neural network language model. Dans *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5528–5531.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. et Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L.

Bottou, M. Welling, Z. Ghahramani, et K. Q. Weinberger (dir.), *Advances in Neural Information Processing Systems 26* 3111–3119. Curran Associates, Inc.

Miller, G. A. (1995). Wordnet : A lexical database for english. *Commun. ACM*, *38*(11), 39–41.

Mitchell, J. et Lapata, M. (2008). Vector-based models of semantic composition. Dans *proceedings of ACL-08 : HLT*, 236–244.

Moher, M. (1993). Decoding via cross-entropy minimization. Dans *Proceedings of GLOBECOM '93. IEEE Global Telecommunications Conference*, 809–813 vol.2.

Mori, K., Jatowt, A. et Ishizuka, M. (2003). Enhancing conversational flexibility in multimodal interactions with embodied lifelike agent. Dans *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, 270–272., New York, NY, USA. ACM.

Müller, M. et Volk, M. (2013). *Statistical Machine Translation of Subtitles : From OpenSubtitles to TED*, Dans I. Gurevych, C. Biemann, et T. Zesch (dir.). *Language Processing and Knowledge in the Web : 25th International Conference, GSCL 2013, Darmstadt, Germany, September 25-27, 2013. Proceedings*, (p. 132–138). Springer Berlin Heidelberg : Berlin, Heidelberg.

Nallapati, R., Zhou, B., dos Santos, C. N., Gülçehre, Ç. et Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. Dans *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, 280–290.

Navigli, R. et Ponzetto, S. P. (2012). Babelnet : The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193*, 217–250.

Noseworthy, M. (2018). *Evaluation of Neural Dialogue Models in Large Domains*. (Thèse de doctorat). McGill University Libraries.

Olabiyi, O., Khazane, A. et Mueller, E. T. (2018). A persona-based multi-turn conversation model in an adversarial learning framework. Dans *17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, December 17-20, 2018*, 489–494.

Pang, B., Nijkamp, E., Han, W., Zhou, L., Liu, Y. et Tu, K. (2020). Towards holistic and automatic evaluation of open-domain dialogue generation. Dans *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3619–3629., Online. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T. et Zhu, W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. Dans *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Park, Y., Cho, J. et Kim, G. (2018). A hierarchical latent structure for variational conversation modeling. Dans *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, 1792–1801.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A. et Shlens, J. (2019). Stand-alone self-attention in vision models. Dans *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 68–80.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A. et Tran, D. (2018). Image transformer. Dans J. Dy et A. Krause (dir.). *Proceedings of the 35th International Conference on Machine Learning*, volume 80 de *Proceedings of Machine Learning Research*, 4055–4064. PMLR.

Pennington, J., Socher, R. et Manning, C. D. (2014). Glove : Global vectors for word representation. Dans *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Pietquin, O. et Hastie, H. (2013a). A survey on metrics for the evaluation of user simulations. *Knowledge Engineering Review*, *28*(01), 59–73. first published as FirstView.

Pietquin, O. et Hastie, H. (2013b). A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, *28*(1), 59–73.

Popović, M. et Ney, H. (2007). Word error rates : Decomposition over pos classes and applications for error analysis. Dans *Proceedings of the Second Workshop on Statistical Machine Translation*, 48–55. Association for Computational Linguistics.

Qiu, M., Li, F.-L., Wang, S., Gao, X., Chen, Y., Zhao, W., Chen, H., Huang, J. et Chu, W. (2017). Alime chat : A sequence to sequence and rerank based chatbot engine. Dans *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, volume 2, 498–503.

Radford, A., Narasimhan, K., Salimans, T. et Sutskever, I. (2018). *Improving language understanding with unsupervised learning*. Rapport technique.

Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M. et Sutskever, I. (2019a). Better language models and their implications. *OpenAI Blog https ://openai. com/blog/better-language-models*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et Sutskever, I. (2019b). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Ritter, A., Cherry, C. et Dolan, B. (2010). Unsupervised modeling of twitter conversations. Dans *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, 172–180., Stroudsburg, PA, USA. Association for Computational Linguistics.

Ritter, A., Cherry, C. et Dolan, W. B. (2011). Data-driven response generation in social media. Dans *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, 583–593., Stroudsburg, PA, USA. Association for Computational Linguistics.

Robbins, H. et Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rus, V. et Lintean, M. (2012a). A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. Dans *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, 157–162., Montréal, Canada. Association for Computational Linguistics.

Rus, V. et Lintean, M. (2012b). A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. Dans *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, 157–162. Association for Computational Linguistics.

Rush, A. M., Chopra, S. et Weston, J. (2015). A neural attention model for abstractive sentence summarization. Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 379–389.

Sankar, C. (2020). *Neural approaches to dialog modeling*. (Thèse de doctorat). University of montreal.

Sato, S., Akama, R., Ouchi, H., Suzuki, J. et Inui, K. (2020). Evaluating dialogue generation systems via response selection. Dans *ACL 2020*, volume 27, 677–681.

Schuetzler, R. M. (2015). *Dynamic Interviewing Agents : Effects on Deception, Nonverbal Behavior, and Social Desirability*. The University of Arizona.

Secic, A., Krpan, M. et Kuzle, I. (2019). Vibro-acoustic methods in the condition assessment of power transformers : A survey. *IEEE Access*, *7*, 83915–83931.

Sennrich, R. (2012). Perplexity minimization for translation model domain adaptation in statistical machine translation. Dans *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 539–549. Association for Computational Linguistics.

Serban, I., Klinger, T., Tesauro, G., Talamadupula, K., Zhou, B., Bengio, Y. et Courville, A. (2017a). Multiresolution recurrent neural networks : An application to dialogue response generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

Serban, I., Sordoni, A., Bengio, Y., Courville, A. et Pineau, J. (2016a). Building end-to-end dialogue systems using generative hierarchical neural network models. volume 30.

Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A. et Bengio, Y. (2017b). A hierarchical latent variable encoder-decoder model for generating dialogues. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C. et Pineau, J. (2016b). Building end-to-end dialogue systems using generative hierarchical neural network models. Dans *AAAI*, 3776–3784.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A. et Bengio, Y. (2017c). A hierarchical latent variable encoder-decoder model for generating dialogues. Dans *Thirty-First AAAI Conference on Artificial Intelligence*.

Shang, L., Sakai, T., Lu, Z., Li, H., Higashinaka, R. et Miyao, Y. (2016). Overview of the ntcir-12 short text conversation task. Dans *the 12th NTCIR Conference on Evaluation of Information Access Technologies*.

Shazeer, N. et Stern, M. (2018). Adafactor : Adaptive learning rates with sublinear memory cost. Dans J. Dy et A. Krause (dir.). *Proceedings of the 35th International Conference on Machine Learning*, volume 80 de *Proceedings of Machine Learning Research*, 4596–4604. PMLR.

Shiv, V. L. et Quirk, C. (2019). Novel positional encodings to enable tree-based transformers. Dans *The international Conference on Neural Information Processing Systems (NIPS)*, 12058–12068.

Shuster, K., Humeau, S., Hu, H., Bordes, A. et Weston, J. (2019). Engaging image captioning via personality. Dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simard, P., Steinkraus, D. et Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. Dans *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 958–963.

Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C. et Lopez, A. (2013). Dirt cheap web-scale parallel text from the Common Crawl. Dans *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 1374–1383., Sofia, Bulgaria. Association for Computational Linguistics.

Sohoni, N. S., Aberger, C. R., Leszczynski, M., Zhang, J. et Ré, C. (2019). Low-memory neural network training : A technical report.

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J. et Nie, J.-Y. (2015a). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. Dans *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, 553–562., New York, NY, USA. ACM.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J., Gao, J. et Dolan, B. (2015b). A neural network approach to context-sensitive generation of conversational responses. Dans *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, 196–205.

Streiner, D. L. et Norman, G. R. (2006). "precision" and "accuracy" : Two terms that are neither. *Journal of Clinical Epidemiology*, *59*(4), 327–330.

Sukhbaatar, S. et Fergus, R. (2015). Learning from noisy labels with deep neural networks. Dans *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Sukhbaatar, S., Grave, E., Bojanowski, P. et Joulin, A. (2019). Adaptive attention span in transformers. Dans *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1 : Long Papers*, 331–335.

Surmenok, P. (2016). Chatbot architecture (2016). Récupéré de `http://pavel.surmenok.com/2016/09/11/chatbot-architecture/`

Sutskever, I., Vinyals, O. et Le, Q. V. (2014). Sequence to sequence learning with neural networks. Dans *Advances in neural information processing systems*, 3104–3112.

Tan, R., Sun, J., Su, B. et Liu, G. (2019). Extending the transformer with context and multi-dimensional mechanism for dialogue response generation. Dans *CCF International Conference on Natural Language Processing and Chinese Computing*, 189–199.

Tang, D., Qin, B. et Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. Dans *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1422–1432.

Tay, Y., Dehghani, M., Aribandi, V., Gupta, J. P., Pham, P., Qin, Z., Bahri, D., Juan, D. et Metzler, D. (2021). Omninet : Omnidirectional representations from transformers. Dans *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, 10193–10202.

Tian, Z., Yan, R., Mou, L., Song, Y., Feng, Y. et Zhao, D. (2017). How to make context more useful ? an empirical study on context-aware neural conversational models. Dans *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 231–236., Vancouver, Canada. Association for Computational Linguistics.

Tong, Y. L. X. et Yen, C.-M. (2014). Variational neural conversational model. Dans *The 31st International Conference on Machine Learning (ICML 2014)*.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P. et Suleman, K. (2017). Newsqa : A machine comprehension dataset. Dans *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, 191–200.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. et Polosukhin, I. (2017). Attention is all you need. Dans *Advances in neural information processing systems*, 5998–6008.

Verspoor, K. et Cohen, K. B. (2013). *Natural Language Processing*, Dans W. Dubitzky, O. Wolkenhauer, K.-H. Cho, et H. Yokota (dir.). *Encyclopedia of Systems Biology*, (p. 1495–1498). Springer New York : New York, NY

Vinyals, O. et Le, Q. V. (2015). A neural conversational model. Dans *The 32nd International Conference on Machine Learning (ICML) Deep Learning Workshop*.

Wang, C. et Jiang, H. (2019). The lower the simpler : Simplifying hierarchical recurrent models. Dans *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4005–4009.

Wang, H., Lu, Z., Li, H. et Chen, E. (2013). A dataset for research on short-text conversations. Dans *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 935–945.

Wang, X. et Yuan, C. (2016). Recent advances on human-computer dialogue. *Caai Transactions on Intelligence Technology*, *1*(4), 303–312.

Wang, Y., Ke, P., Zheng, Y., Huang, K., Jiang, Y., Zhu, X. et Huang, M. (2021). A large-scale chinese short-text conversation dataset. Dans *CCF International Conference on Natural Language Processing and Chinese Computing*, 91–103. Springer.

Wei, B., Lu, S., Mou, L., Zhou, H., Poupart, P., Li, G. et Jin, Z. (2019). Why do neural dialog systems generate short and meaningless replies ? a comparison between dialog and translation. Dans *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7290–7294.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Weston, J., Bordes, A., Chopra, S., Mikolov, T., Rush, A. et van Merrienboer, B. (2015). *Towards AI-Complete Question Answering : A Set of Prerequisite Toy Tasks*. Rapport technique, Facebook Research.

Wical, K. (2002). *Document knowledge base research and retrieval system*. Rapport technique, Google Patents. US Patent 6,460,034.

Widrow, B. et Lehr, M. A. (1990). 30 years of adaptive neural networks : perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, *78*(9), 1415–1442.

Wilcox, B., Wilcox, S., Psych, B. et Arts, D. F. (2010). Suzette, the most human computer.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. et Rush, A. (2020). Transformers : State-of-the-art natural language processing. Dans *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, 38–45., Online. Association for Computational Linguistics.

Woudenberg, A. v. (2014). *A Chatbot Dialogue Manager-Chatbots and Dialogue Systems : A Hybrid Approach*. (Mémoire de maîtrise). Open Universiteit Nederland.

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N. et Auli, M. (2019). Pay less attention with lightweight and dynamic convolutions. Dans *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Xia, P., Wu, S. et Durme, B. V. (2020). Which *bert ? A survey organizing contextualized encoders. Dans B. Webber, T. Cohn, Y. He, et Y. Liu (dir.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 7516–7533. Association for Computational Linguistics.

Xing, C., Wu, Y., Wu, W., Huang, Y. et Zhou, M. (2018). Hierarchical recurrent attention network for response generation. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Xiong, C., Merity, S. et Socher, R. (2016). Dynamic memory networks for visual and textual question answering. Dans *International conference on machine learning*, 2397–2406.

Yan, Z., Duan, N., Bao, J., Chen, P., Zhou, M., Li, Z. et Zhou, J. (2016). Docchat : An information retrieval approach for chatbot engines using unstructured documents. Dans *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 516–525.

Zanettin, F. (2014). *Translation-driven corpora : Corpus resources for descriptive and applied translation studies*. Routledge.

Zhang, B., Xiong, D., Xie, J. et Su, J. (2020a). Neural machine translation with gru-gated attention model. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4688–4698.

Zhang, H., Lan, Y., Pang, L., Chen, H., Ding, Z. et Yin, D. (2020b). Modeling topical relevance for multi-turn dialogue generation. Dans C. Bessiere (dir.). *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 3737–3743. ijcai.org.

Zhang, H., Lan, Y., Pang, L., Guo, J. et Cheng, X. (2019a). Recosa : Detecting the relevant contexts with self-attention for multi-turn dialogue generation. Dans A. Korhonen, D. R. Traum, et L. Màrquez (dir.). *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1 : Long Papers*, 3721–3730. Association for Computational Linguistics.

Zhang, H., Shao, J. et Salakhutdinov, R. (2019b). Deep neural networks with multi-branch architectures are intrinsically less non-convex. Dans K. Chaudhuri et M. Sugiyama (dir.). *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 de *Proceedings of Machine Learning Research*, 1099–1109. PMLR.

Zhang, N. (2020). Learning adversarial transformer for symbolic music generation. *IEEE Transactions on Neural Networks and Learning Systems*, 1–10.

Zhang, Y., Sun, S., Galley, M., Chen, Y., Brockett, C., Gao, X., Gao, J., Liu, J. et Dolan, B. (2020c). DIALOGPT : Large-scale generative pre-training for conversational response generation. Dans A. Celikyilmaz et T. Wen (dir.). *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics : System Demonstrations, ACL 2020, Online, July 5-10, 2020*, 270–278. Association for Computational Linguistics.

Zhao, T., Zhao, R. et Eskénazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. Dans R. Barzilay et M. Kan (dir.). *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1 : Long Papers*, 654–664. Association for Computational Linguistics.

Zhou, H., Ke, P., Zhang, Z., Gu, Y., Zheng, Y., Zheng, C., Wang, Y., Wu, C. H., Sun, H., Yang, X., Wen, B., Zhu, X., Huang, M. et Tang, J. (2021). EVA : an open-domain chinese dialogue system with large-scale generative pre-training. *CoRR*, *abs/2108.01547*.

Zhou, J., Cao, Y., Wang, X., Li, P. et Xu, W. (2016). Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*, *4*, 371–383.

Zhou, W., Michel, W., Irie, K., Kitza, M., Schlüter, R. et Ney, H. (2020). The rwth asr system for ted-lium release 2 : Improving hybrid hmm with specaugment. Dans *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7839–7843.

Zhou, X. et Wang, W. Y. (2018). Mojitalk : Generating emotional responses at scale. Dans I. Gurevych et Y. Miyao (dir.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1 : Long Papers*, 1128–1137. Association for Computational Linguistics.

Ziemski, M., Junczys-Dowmunt, M. et Pouliquen, B. (2016). The United Nations parallel corpus v1.0. Dans *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 3530–3534., Portorož, Slovenia. European Language Resources Association (ELRA).