Systèmes de dialogue apprenant tout au long de leur vie : de l'élaboration à l'évaluation

Mathilde Veron

université PARIS-SACLAY

Systèmes de dialogue apprenant tout au long de leur vie : de l'élaboration à l'évaluation

Lifelong Learning Dialogue Systems: from Conception to Evaluation

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique, Référent : Faculté des sciences d'Orsay

Thèse préparée au **Laboratoire Interdisciplinaire des Sciences du Numérique** (**Université Paris-Saclay**, **CNRS**) et au **Laboratoire National de Métrologie et d'Essais**, sous la direction de **Sophie ROSSET**, directrice de recherche et sous la co-supervision de **Olivier GALIBERT**, ingénieur-docteur et de **Guillaume BERNARD**, ingénieur-docteur.

Thèse soutenue à Paris-Saclay, le 15 décembre 2022, par

Mathilde VERON

Composition du jury

Membres du jury avec voix délibérative

Bich-Liên DOAN Professeure, LISN, Université Paris-Saclay	Présidente
Yannick ESTÈVE Professeur, LIA, Avignon Université	Rapporteur & Examinateur
François PORTET Professeur, LIG, Université Grenoble Alpes	Rapporteur & Examinateur
Frédéric BÉCHET Professeur, LIS, Aix Marseille Université	Examinateur
Chloé CLAVEL Professeure, LTCI, Télécom Paris-Tech	Examinatrice

THESE DE DOCTORAT
NNT : 2022UPASG089

Remerciements

Il y a 4 ans, si quelqu'un m'avait dit que je serai docteure en informatique plus tard, je lui aurais ri au nez. A l'époque j'avais de grands a priori sur ce qu'était la recherche et encore plus sur ce que cela signifiait d'être doctorant. Je m'imaginais alors totalement incapable de faire une thèse, que ce n'était pas pour moi et que je n'avais pas les compétences nécessaires. 4 ans plus tard, je suis donc fière d'avoir pu me prouver le contraire, je suis fière d'être allée au bout de ce grand défi et fière du chemin que j'ai parcouru. Évidemment je ne dirais pas que la route a été facile ni qu'elle n'a jamais été semée d'embûches, j'ai souvent remis en question les questions de recherche auxquelles je m'intéressais ainsi que mes choix expérimentaux. Cependant, grâce à ça j'ai beaucoup appris sur moi-même et ces dernières années ont été très enrichissantes, tant du point de vue des connaissances et des compétences que j'ai pu acquérir, que d'un point de vue humain. Il est donc important pour moi de remercier les personnes qui m'ont aidée tout au long de ce chemin.

Tout d'abord je tiens à remercier sincèrement ma directrice de thèse, merci Sophie d'avoir cru en moi alors que moi-même je n'y croyais pas. Je me rappelle encore des premiers mois où j'ai pu travailler à tes côtés et où tu as su voir que j'avais le potentiel de faire une thèse alors que je n'y connaissais rien, ni en traitement automatique des langues, ni en apprentissage profond. Je remercie aussi chaleureusement Olivier et Guillaume, mes encadrants de thèse côté entreprise, merci pour votre bienveillance et votre soutien. Merci de m'avoir permis de garder les pieds sur terre et de me concentrer sur ce qui était important. Merci à tous les trois de toujours vous être montrés compréhensif et de m'avoir encouragée sans me mettre la pression.

Je souhaite aussi remercier tous les doctorants dont j'ai pu croiser la route. Merci pour ces discussions, tantôt autour d'un repas, tantôt autour d'un thé, aussi bien à la cafet' que dans les bureaux, dans les couloirs ou même à un bar. Merci pour ces discussions qui rassurent ("Ouf je ne suis pas la seule"), ces discussions où on apprend des choses ("Ahhh c'est à ça que ça sert, je pourrai essayer d'ajouter ça dans mes expériences...") et merci pour tous ces moments d'entraide. Merci aussi

aux permanents du LISN pour vos précieux conseils et pour le temps que vous m'avez consacré.

Merci évidemment aussi à mes collègues côté entreprise. Merci pour le soutien et vos conseils et plus largement pour l'ambiance générale qui m'a permis de voir en le LNE un endroit où il était agréable de travailler.

Plus largement je souhaite remercier le LNE pour son soutien financier et matériel tout au long de ma thèse, notamment avec l'aide de l'ANRT (contrat CIFRE # 2019/0628). Je remercie aussi le LISN pour m'avoir accueilli au sein de ses locaux et d'avoir permis tous ces échanges avec les autres doctorants et permanents du laboratoire mais aussi pour m'avoir permis de participer au projet européen LIHLITH (ANR-17-CHR2-0001-03). Je remercie aussi le QU-Lab à Berlin pour son accueil et les échanges que j'ai pu avoir lors de mon séjour doctoral dans ses locaux. Les expériences réalisées au cours de la thèse ont pu se faire grâce aux plateformes de calcul Saclay-IA et Jean Zay (GENCI-IDRIS) via les dossiers AD011012609 et AD011012609R1. Je remercie ainsi toutes les personnes associées à chacune de ces structures côté administratif et ingénierie.

Enfin je souhaite remercier mes proches et ma famille. Merci pour votre soutien ces 3 dernières années et pour vos paroles d'encouragement. En particulier je tiens à remercier mon mari, merci de m'avoir écoutée et de m'avoir encouragée lorsque j'avais l'impression que je n'y arriverais pas.

Ainsi merci à tous, je vous souhaite de vous épanouir quels que soient vos objectifs et d'aller au bout des projets qui vous tiennent à cœur. Bonne continuation !

Table des matières

Chapitre 1

Introduction

1.1 Motivations

Qu'est-ce qu'un système de dialogue ?

Charlie veut réserver un billet de train. Charlie va donc sur son moteur de recherche préféré, tape « réserver un train » et clique sur le premier lien qui s'affiche. À peine la page chargée, une petite fenêtre s'ouvre sur le côté, un visage sympathique à l'allure de robot apparaît et indique « Besoin d'aide ? Discutons ensemble pour trouver une solution ». À l'image de Charlie, il est aujourd'hui de plus en plus fréquent de croiser la route d'un chatbot dans le cadre d'une recherche sur internet. Les chatbots sont des systèmes automatisés qui permettent à leurs utilisateurs de réaliser des tâches définies et d'obtenir certaines informations dans le cadre d'une conversation. Beaucoup d'entreprises se sont aujourd'hui munies d'un tel chatbot afin d'améliorer leur relation client. De fait, un chatbot est disponible 24 heures sur 24 et permet de gérer les questions et les tâches les plus fréquentes, qui sont d'ailleurs souvent les moins valorisantes pour un opérateur humain. Ainsi, depuis 2016, la SNCF s'est dotée d'un chatbot dans le but d'aider ses clients à réaliser un certain nombre de tâches [1] : rechercher un trajet en train ou en bus, réserver et payer ses billets, recevoir des informations sur sa réservation, etc. Les chatbots offrent en théorie une interface plus intuitive et facile d'utilisation, permettant à leurs utilisateurs d'éviter des recherches sur plusieurs page web et/ou d'avoir à se familiariser avec des interfaces de recherche parfois complexes. En effet, puisque les interactions se font en langage naturel, par écrit ou bien à l'oral, chaque utilisateur est en mesure de formuler sa demande avec ses propres mots. Les interactions se faisant sur plusieurs tours de parole, l'utilisateur peut compléter ses requêtes au

1. https://vous.sncf-connect.com/article/focus/ouibot-le-compagnon-de-voyage-30

fil de l'eau, parfois à la demande du chatbot lui-même. Le système doit alors être capable de garder en mémoire les informations fournies par l'utilisateur au cours des tours précédents afin de réagir de manière adéquate à l'énoncé utilisateur du tour courant. L'ensemble de ces tours de parole constituent ce que l'on appelle un dialogue. Dans le domaine de la recherche, nous parlons plus spécifiquement de *systèmes de dialogue*, le terme *chatbot* étant plus général et couvrant d'autres types d'agents conversationnels (voir section 2.2.1). Pour être plus précis, les systèmes de dialogue décrits dans ce paragraphe sont dits *orientés tâche* et nous nous concentrons dans ce mémoire sur ce type de système.

Limitations des systèmes de dialogue actuels.

Malgré l'adoption des systèmes de dialogue par de nombreuses entreprises, ceux-ci souffrent aujourd'hui d'un certain nombre de limitations, ce qui explique que leur étude constitue encore un sujet de recherche important.

Depuis les années 2010, de nombreux progrès ont été réalisés conjointement en apprentissage profond et en Traitement Automatique des Langues (TAL) [Krizhevsky et al., 2012, Ruder, 2018, Devlin et al., 2019a, Raffel et al., 2020], jusqu'à l'adoption générale des réseaux de neurones dans ce dernier domaine, appliqués notamment aux systèmes de dialogue [Ni et al., 2022]. Cependant, l'entraînement d'un modèle neuronal nécessite des données d'entraînement, dont la quantité et la qualité influent directement sur les capacités et les performances d'un système reposant sur un tel modèle. Dans le cadre des systèmes de dialogue, un modèle entraîné peut correspondre à un module du système (par exemple le module de compréhension de la langue, ou NLU en anglais), ou au système complet dans le cas d'une approche bout-à-bout (*end-to-end* en anglais). Dans la majeure partie des cas, les données d'entraînement doivent aussi comporter des annotations réalisées par des experts humains. Par exemple dans le cadre de la détection de l'intention utilisateur pour le NLU, l'énoncé `"J'ai besoin d'un hôtel à Paris pour le week-end du 8 octobre"` peut être annoté avec l'intention `"réservation d'un hôtel"`. La collecte et l'annotation des données d'entraînement consiste ainsi en des opérations coûteuses. Ceci explique en partie que, malgré le fait que tous les modèles à l'état de l'art dédiés aux systèmes de dialogue en recherche reposent sur des réseaux neuronaux, certains chatbots développés par des entreprises sont encore aujourd'hui basés sur des technologies plus simples qui ne nécessitent pas de données d'entraînement. Cette **dépendance aux données d'entraînement** définit une première limitation des systèmes de dialogue.

Une deuxième limitation consiste dans le **manque de flexibilité** de ces systèmes. Ceux-ci sont généralement incapables de généraliser leurs capacités si le type de cas de figure rencontré en production n'a pas été défini lors du développement du

système ou s'il n'a pas été présenté dans les exemples d'entraînement. Parmi ces cas de figure, nous pouvons citer la présence d'une intention inconnue, d'un type de critère de recherche inconnu ou d'un domaine inconnu. Concrètement, nous pouvons parfois observer l'incapacité d'un système à comprendre que l'utilisateur veut mettre fin au dialogue avec un `"Au revoir"` dans un contexte de dialogue précis. Ce peut être parce que cet acte de dialogue et ce contexte de dialogue n'ont jamais été rencontrés en même temps pendant l'entraînement, ou parce que les développeurs n'ont pas implémenté ce cas de figure, alors que le système comprend correctement un `"Au revoir"` dans un autre contexte de dialogue. Nous pouvons aussi être surpris par l'incapacité d'un système de dialogue à fonctionner dans un autre domaine que celui prévu initialement, alors que les tâches à réaliser sont très proches, par exemple si l'on étend la réservation d'un billet de train à celle d'un billet d'avion. Cette limitation est surprenante d'un point de vue humain puisque nous savons facilement identifier des similitudes entre domaines et transférer nos connaissances d'un domaine à un autre.

Cependant, il est généralement impossible de définir à l'avance l'ensemble des cas de figure possibles ou de disposer d'un corpus de données les recouvrant tous. De plus, le monde étant en constante évolution et les besoins des utilisateurs étant amenés à évoluer au cours du temps, les connaissances et les capacités d'un système sont vouées à devenir de plus en plus incomplètes au fur et à mesure du temps. Dans le cas d'un système dédié à la réservation de chambres d'hôtel, celui ci devrait par exemple fonctionner pour des hôtels nouvellement construits, dont les noms n'auront jamais été vus avant le déploiement, ou bien satisfaire des nouveaux besoins utilisateurs comme par exemple répondre à des questions concernant les restrictions sanitaires actuelles, ou plus largement s'adapter à un nouveau vocabulaire. Nous pouvons aussi prendre l'exemple d'une entreprise qui après avoir déployé un système de dialogue en anglais, souhaite déployer ce système en d'autres langues au cours du temps sans avoir à recommencer de zéro l'ensemble du processus de développement réalisé pour le système en anglais. Si les systèmes actuels manquent de flexibilité pour répondre à ces problèmes, il est donc **nécessaire de permettre à ces systèmes de s'adapter à de nouveaux cas de figure au cours du temps.** Lorsque l'on se renseigne sur un système de dialogue déployé par une entreprise, il n'est pas rare de lire « le système s'améliore grâce aux interactions que vous avez avec lui ». Il est ainsi facile de penser que le système est capable de s'améliorer instantanément et de manière autonome. Cependant, de manière générale, cette idée est fausse si le système repose sur un modèle d'apprentissage statistique ou profond, puisque l'entraînement d'un modèle se fait généralement hors ligne et une fois le système déployé, celui-ci n'est pas en mesure de s'améliorer de lui-même au fur et à mesure de ses interactions avec les utilisateurs. L'amélioration du système nécessite d'abord d'identifier les points

pouvant être améliorés – souvent via l'analyse de journaux d'événements –, ensuite de collecter et d'annoter des nouvelles données, puis d'entraîner ou d'adapter le modèle sur ces nouvelles données, pour finalement déployer et rendre accessible aux utilisateurs la version améliorée du chatbot. Aujourd'hui, ces étapes sont réalisées par des personnes qualifiées et l'amélioration d'un chatbot consiste en une opération fastidieuse et coûteuse. La **difficulté d'adapter les systèmes de dialogue au cours du temps** consiste ainsi en une troisième limitation.

Lifelong learning et apprentissage sur le terrain.

Idéalement, nous souhaiterions que les systèmes de dialogue soient capables d'apprendre de nouvelles connaissances au cours du temps et de profiter des connaissances apprises précédemment pour apprendre plus facilement et plus efficacement de nouvelles connaissances (notion de transfert) tout en gardant de bonnes performances sur l'ensemble de ses connaissances, c'est à dire de minimiser l'oubli. Cette capacité d'apprentissage au cours du temps a été introduite par [Thrun and Mitchell, 1995] sous le nom de *Lifelong Learning* en anglais (LL), qui peut être traduit en français comme l'apprentissage tout au long de sa vie. Après l'essor de l'apprentissage profond (*deep learning* en anglais), ont été employés pour décrire cette capacité à la fois le terme de « Lifelong Learning » mais aussi celui d' « apprentissage continu » (*continual learning* en anglais) [Parisi et al., 2019]. Cependant, tandis que le LL peut être appliqué à n'importe quel système, l'apprentissage continu n'est lui applicable qu'aux modèles d'apprentissage profond. De plus, la définition du LL a elle-même évolué au cours des années jusqu'à incorporer explicitement l'idée d'apprentissage sur le terrain (*on-the-job learning* en anglais) [Liu, 2020]. L'apprentissage sur le terrain suppose que l'apprentissage du système, ou du moins son adaptation, se fait après le déploiement du système, en continu et surtout en autonomie. Un système capable d'apprendre sur le terrain doit ainsi être en mesure, sans l'aide d'expert et au cours de son utilisation, de (1) détecter qu'un nouvel élément peut être appris, (2) récupérer et identifier le nouvel élément et (3) s'adapter à ce nouvel élément. Dans le cas où le système à adapter repose sur un modèle d'apprentissage, la deuxième étape consiste en la collecte et l'annotation de nouveaux exemples d'entraînement. Cette étape devra donc être réalisée par le système lui-même.

Dans le cadre de l'apprentissage continu au contraire, l'entraînement du modèle se fait hors ligne, de manière séquentielle, et sur des nouvelles données annotées disponibles à chaque étape de la séquence. Il n'est donc pas nécessaire dans le cas de l'apprentissage continu de réaliser les étapes (1) et (2) de l'apprentissage sur le terrain. Cependant, dans le cadre d'un système reposant sur de l'apprentissage profond, l'étape (3) pourra se faire à l'aide de techniques utilisées pour

l'apprentissage continu afin de maximiser le transfert en avant tout en minimisant l'oubli.

Dans le cadre des systèmes de dialogue, de manière similaire aux êtres humains, un système de dialogue capable d'apprendre tout au long de sa vie pourrait ainsi profiter des interactions avec son environnement, notamment avec ses utilisateurs, afin d'améliorer ses performances au cours du temps. Un système de dialogue pourrait notamment prendre en compte les retours de ses utilisateurs comme par exemple face à une erreur de compréhension (`"Quoi? Mais ce n'est pas du tout ce que j'ai demandé !"`).

De nombreux paradigmes déjà définis, notamment dans le domaine de l'apprentissage profond, sont liés au LL et leur étude permet ainsi en partie l'étude du LL. Parmi ces paradigmes, nous pouvons citer l'apprentissage continu comme vu précédemment, mais aussi l'apprentissage zero-shot, l'apprentissage par renforcement et l'apprentissage en ligne. Les définitions de ces paradigmes ainsi que leurs différences avec le LL sont décrites en section 3.2.

1.2 Contexte

La thèse a été réalisée dans le cadre d'un contrat CIFRE entre le Laboratoire Interdisciplinaire des Sciences du Numérique [2] (LISN) et le Laboratoire National de Métrologie et d'Essais [3] (LNE), respectivement au sein du département des Sciences et Technologies des Langues et du département de l'Évaluation de l'Intelligence Artificielle.

La thèse s'est partiellement déroulée dans le cadre du projet CHIST-ERA LIHLITH [4].

1.3 Questions de recherche et structure du mémoire

Motivés par les limitations des systèmes de dialogue exposées en section 1.1, nous présentons dans ce mémoire des travaux en lien avec les questions de recherche suivantes :

— *Comment définir un système de dialogue apprenant tout au long de sa vie ?*

2. Université Paris-Saclay, CNRS, 91400, Orsay, France, https://www.lisn.upsaclay.fr/
3. 78197, Trappes, France, https://www.lne.fr/fr
4. Learning to Interact with Humans by Lifelong Interaction with Humans, http://ixa.ehu.eus/lihlith/

— *Comment construire un tel système ? Quelles techniques et méthodologies peuvent être employées ?*
— *Comment évaluer un tel système ?*

Pour répondre à ces questions, nous commençons par présenter dans le chapitre 2 des éléments généraux permettant d'appréhender le contexte des contributions présentées dans ce mémoire. Ainsi, en section 2.1 nous réalisons un historique synthétique des avancées récentes dans le domaine du traitement automatique des langues (TAL) et présentons le mécanisme d'attention ainsi que plusieurs modèles de langue pré-entraînés couramment utilisés en TAL. Par la suite, nous présentons en section 2.2 les systèmes de dialogue orientés tâche. Pour cela, nous commençons par définir plusieurs types de systèmes de dialogue et expliquons en quoi ils se différencient des systèmes de dialogue orientés tâche. Ensuite, nous décrivons les modules qui composent généralement un système de dialogue orienté tâche et donnons un aperçu des approches utilisées pour permettre leur fonctionnement. Enfin en section 2.3, nous discutons l'état de l'art dans les domaines du TAL et des systèmes de dialogue puis expliquons nos motivations vis-à-vis du Lifelong Learning (LL).

Par la suite, nous décrivons le LL dans le chapitre 3. En section 3.1, après avoir retracé l'historique de ce paradigme, nous commençons par donner notre définition du LL en présentant les cinq caractéristiques associées à notre définition, puis nous appliquons le LL aux systèmes de dialogue orientés tâche. Ensuite, en section 3.2 nous présentons plusieurs paradigmes en lien avec le LL. Puis en section 3.3 nous présentons deux systèmes opérationnels en lien avec le LL. Enfin, en section 3.4 nous discutons de l'intérêt du LL malgré sa complexité.

Dans un premier temps, nous choisissons dans le chapitre 4 de nous concentrer sur l'application et l'évaluation de l'apprentissage sur le terrain d'un système de dialogue orienté tâche. Comme présenté en section 1.1, l'apprentissage sur le terrain est très proche du LL, bien que moins complexe, ce qui permet d'en faciliter l'étude et la comparaison avec d'autres travaux. Ainsi, en section 4.2 nous présentons plusieurs systèmes de dialogue en lien avec l'apprentissage sur le terrain et la manière dont ils sont évalués, puis nous présentons des méthodologies d'évaluation liées à l'apprentissage sur le terrain. Ensuite, en section 4.3 nous présentons notre méthodologie générale pour l'évaluation de l'apprentissage sur le terrain d'un système de dialogue. De manière à tester notre méthodologie d'évaluation, nous présentons en section 4.4 un système de dialogue capable d'améliorer sur le terrain sa détection des slots, puis nous décrivons en section 4.5 la configuration de nos expériences. Enfin, en section 4.6 nous donnons les résultats de nos expériences sur notre système de dialogue dont l'apprentissage sur le terrain est évalué à l'aide de la méthodologie définie au préalable. Nous discutons aussi les résultats et la

méthodologie d'évaluation.

Dans un second temps, nous nous concentrons dans le chapitre 5 sur deux aspects du LL non étudiés dans le chapitre précédent qui consistent en l'étude du transfert et en l'application de l'apprentissage continu. Ainsi nous étudions le transfert inter-langues sur la tâche de détection des slots dans un cadre novateur qui est l'apprentissage continu d'une séquence de langues. Pour cela, nous réalisons en section 5.2 un état de l'art à la fois sur l'apprentissage continu général, sur l'apprentissage continu appliqué aux systèmes de dialogue et sur les études de transfert entre langues existantes. Ensuite, en section 5.3, nous présentons les configurations générales de nos expériences puis définissons en section 5.4 les différentes mesures de transfert utilisées dans ce chapitre. En section 5.5, nous présentons nos résultats en matière de transfert et comparons les mesures de transfert obtenues dans le cadre de l'apprentissage continu à celles obtenues dans d'autres contextes expérimentaux. Par la suite, nous cherchons en section 5.6 à analyser les résultats de transfert continu en considérant davantage la séquence d'entraînement, puis en section 5.7 nous nous intéressons aux capacités d'un modèle entraîné de manière continue. Enfin en section 5.8, nous réalisons d'abord une synthèse des résultats obtenus dans le cadre de nos expériences, puis nous proposons une nouvelle direction de recherche visant à combiner l'apprentissage continu et l'apprentissage actif afin de se rapprocher du LL.

Dans un troisième temps, nous cherchons dans le chapitre 6 à complexifier la tâche étudiée et à intégrer davantage l'aspect dialogique à nos expériences en nous concentrant sur la tâche de suivi de l'état du dialogue. De la même manière que dans le chapitre précédent, nous nous intéressons à l'étude du transfert mais nous complexifions de nouveau les expériences en étudiant l'adaptation d'un modèle à de nouveaux types de slots jamais vu pendant l'entraînement. Nous nous intéressons donc dans ce chapitre à l'étude du transfert entre domaines, chaque domaine comprenant des slots qui lui sont propres, et réalisons cette étude dans le cadre de l'apprentissage zero-shot afin de diversifier les contextes expérimentaux. Ainsi en section 6.2, nous réalisons un état de l'art des travaux sur la tâche de suivi de l'état du dialogue qui s'intéressent aux capacités de généralisation et de transfert de leur modèle. Par la suite, nous présentons en section 6.3 le corpus de données MultiWOZ [Budzianowski et al., 2018a], puis en section 6.4 le modèle SUMBT [Lee et al., 2019] sur lesquels reposent nos expériences. En section 6.5 nous présentons d'abord les résultats de nos expériences réalisées sur l'étude des capacités de généralisation et de transfert de SUMBT dans le cadre de l'application du modèle à de nouvelles valeurs de slot. En section 6.6 nous présentons ensuite les résultats de nos expériences sur l'étude de transfert inter-domaines dans le cadre de l'apprentissage zero-shot d'un nouveau domaine. Nous présentons notamment une

Chapitre	Type d'apprentissage	Tâche	Nouveauté	Contributions
4	sur le terrain	détection des slots (*slot-filling*)	contexte (patrons de phrases et valeurs de slot)	méthodologie d'évaluation, système de dialogue
5	continu	détection des slots (*slot-filling*)	langue	étude du transfert, fast recovery
6	zero-shot	suivi du dialogue	domaine	étude du transfert, modification de l'architecture d'un modèle existant, modulation de l'attention

TABLEAU 1.1 – Résumé du contenu des chapitres principaux.

méthode appelée *modulation de l'attention* et proposons des variantes au modèle SUMBT afin d'améliorer le transfert dans ce cadre expérimental.

Enfin, dans le chapitre 7, nous commençons en section 7.1 par résumer des différentes conclusions des chapitres principaux, puis nous présentons en section 7.2 nos perspectives concernant le LL et les systèmes de dialogue.

Ainsi la première contribution de ce mémoire consiste en la définition du LL et de son application aux systèmes de dialogue. Les autres contributions sont résumées dans le tableau 1.1 et sont associées à des expériences.

Remarques. La grande majorité des travaux réalisés dans les domaines associés au présent mémoire sont rédigés en anglais. Ainsi, tout au long du mémoire, les termes importants sont indiqués à la fois en français et en anglais. De la même manière, les abréviations en anglais sont souvent préférées afin d'éviter les ambiguïtés et dans le but d'utiliser celles qui sont le plus communément rencontrées dans les travaux scientifiques. En fin de mémoire, un dictionnaire des acronymes et un glossaire sont disponibles. Ils définissent les termes et les acronymes utilisés dans ce mémoire [5]. Le contenu de la majorité des schémas et tableaux est aussi présenté en anglais pour des questions de praticité et de lisibilité puisque l'anglais est souvent plus compact que le français.

5. Dans la version électronique, lorsqu'ils sont utilisés dans le corps du mémoire, les définitions des acronymes et entrées du glossaire sont accessibles directement en cliquant sur les mots correspondant (lien hypertexte).

1.4 Publications

Plusieurs contributions décrites dans ce mémoire ont fait l'objet d'articles. Ces articles ont été relus (*peer review*) et acceptés dans des conférences ou workshops comme listé ci-dessous :

Chapitre 3

— Mathilde Veron, Sahar Ghannay, Anne-Laure Ligozat et Sophie Rosset. Lifelong learning and task-oriented dialogue system: what does it mean?. 2019. In *Proceedings of the Tenth International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, Siracusa, Italy.

— Mathilde Veron. 2019. Lifelong learning et systèmes de dialogue : définition et perspectives (Lifelong learning and dialogue system : definition and discussion). In *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles (TALN) PFIA 2019. Volume III : RECITAL*, pages 563–576, Toulouse, France. ATALA.

Chapitre 4

— Mathilde Veron, Sophie Rosset, Olivier Galibert, Guillaume Bernard. 2020. Evaluate on-the-job learning dialogue systems and a case study for natural language understanding. *Workshop on Human in the Loop Dialogue Systems*, Online. Neural Information Processing Systems.

Chapitre 5

— Juan Manuel Coria, Mathilde Veron, Sahar Ghannay, Guillaume Bernard, Hervé Bredin, Olivier Galibert, and Sophie Rosset. 2022. Analyzing BERT Cross-lingual Transfer Capabilities in Continual Sequence Labeling. In *Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models*, pages 15–25, Virtual. International Conference on Computational Linguistics.

Chapitre 6

— Mathilde Veron, Olivier Galibert, Guillaume Bernard, and Sophie Rosset. 2022. Attention Modulation for Zero-Shot Cross-Domain Dialogue State Tracking. In *Proceedings of the 3rd Workshop on Computational Approaches to Discourse*, pages 86–91, Gyeongju, Republic of Korea and Online. International Conference on Computational Linguistics.

Chapitre 2

Pré-requis, définitions et état de l'art général

Dans ce chapitre, nous présentons des éléments généraux permettant d'appréhender le contexte des contributions présentées dans ce mémoire. Ainsi, en section 2.1 nous réalisons un historique synthétique des avancées récentes dans le domaine du traitement automatique des langues (TAL) et présentons le mécanisme d'attention ainsi que plusieurs modèles de langue pré-entraînés couramment utilisés en TAL. Par la suite, nous présentons en section 2.2 les systèmes de dialogue orientés tâche. Pour cela, nous commençons par définir plusieurs types de systèmes de dialogue et expliquons en quoi ils se différencient des systèmes de dialogue orientés tâche. Ensuite, nous décrivons les modules qui composent généralement un système de dialogue orienté tâche et donnons un aperçu des approches utilisées pour permettre leur fonctionnement. Enfin en section 2.3, nous discutons l'état de l'art dans les domaines du TAL et des systèmes de dialogue puis expliquons nos motivations vis-à-vis du Lifelong Learning (LL).

2.1 Traitement automatique des langues

2.1.1 Historique récent

Dans cette partie nous nous intéressons aux avancées majeures de cette dernière décennie en TAL. Celles-ci sont fortement liées aux avancées réalisées dans le domaine de l'apprentissage profond. Après près de 70 années de recherches sur l'apprentissage profond, un réseau neuronal profond remporte en 2012, pour la première fois, un challenge d'identification d'objets sur des images [Krizhevsky et al., 2012]. Aidé par des avancées technologiques permettant d'augmenter les

capacités de calcul et la création de larges corpus de données annotées, à l'instar d'ImageNet [Deng et al., 2009], l'apprentissage profond gagne alors de plus en plus en popularité et est appliqué à de nombreux domaines, dont le TAL.

Par la suite, [Mikolov et al., 2013] popularise l'utilisation de plongements lexicaux, appelés *word embeddings* en anglais, en proposant avec Word2vec un entraînement neuronal de ces plongements lexicaux ainsi qu'une implémentation efficace, en particulier sur de larges corpus de textes. Les plongements lexicaux ont pour but de représenter les mots comme des vecteurs dans un espace euclidien dense, supposé préserver les relations sémantiques, ce qui rend possible notamment l'approximation de la similarité entre deux mots grâce à des mesures de distance entre leurs représentations respectives. Word2vec repose sur l'hypothèse distributionnelle déjà connue depuis longtemps dans le domaine de la sémantique distributionnelle mais appliquée jusqu'ici seulement à des représentations discrètes. Cette hypothèse stipule que plus des mots sont similaires, plus ils ont de chance d'apparaître dans des contextes similaires, comme dans le cas des mots `"chat"` et `"chien"` qui peuvent être intervertis dans la phrase `"Aujourd'hui j'ai donné une friandise à mon chat/chien."`. Pour ce faire, deux approches différentes ont été implémentées : *continuous bag-of-words* (CBOW) dont l'objectif est de prédire la probabilité d'un mot selon une liste de mots ; et *skip-gram* dont l'objectif est l'inverse de la première. Par la suite, d'autres approches s'intéressent à l'entraînement de plongements lexicaux comme GloVe [Pennington et al., 2014] ou fastText [Joulin et al., 2017]. Ces approches sont des variantes de la tâche de modélisation de la langue [Bengio et al., 2003]. Les représentations vectorielles résultantes, en plus de faciliter l'apprentissage d'un point de vue calcul, améliorent de manière significative les résultats lorsqu'elles sont pré-entraînées sur de larges corpus de données.

En parallèle, plusieurs types de réseaux neuronaux sont appliqués avec succès au TAL, dont : les RNNs (réseaux de neurones récurrents, *recurrent neural networks* en anglais) [Elman, 1990, Hochreiter and Schmidhuber, 1997], avec notamment les LSTM bidirectionnels (*long short-term memory*) [Graves et al., 2013] qui permettent de considérer les mots venant de la droite comme de la gauche dans le cas d'une séquence de mot ; les CNNs (réseaux de neurones convolutionnels) [Kalchbrenner et al., 2014] ; et les réseaux de neurones récursifs qui permettent de considérer le langage de manière hiérarchique [Socher et al., 2013].

[Sutskever et al., 2014] proposent ensuite une architecture générale séquence-vers-séquence (*seq2seq*) afin d'associer une séquence d'éléments à une autre à l'aide des réseaux neuronaux. Cette architecture repose sur un encodeur qui va d'abord traiter chacun des éléments de la séquence d'entrée pour produire une représentation vectorielle. Ensuite, ce vecteur, ainsi qu'un élément indiquant le début de la séquence cible, sont fournis au décodeur pour qu'il génère le premier élément de la

séquence cible. Le décodeur renouvelle ensuite l'étape jusqu'à générer un élément indiquant la fin de la séquence, sachant qu'à chaque étape sont fournis au décodeur le vecteur issu de l'encodeur et la séquence cible résultant de l'étape précédente. Cette architecture est alors appliquée avec succès notamment pour les tâches de traduction.

Les réseaux de neurones mémoriels sont ensuite proposés [Weston et al., 2015], où les états cachés passés du modèle sont gardés en mémoire et où le modèle apprend au cours de l'entraînement quelles informations récupérer dans la mémoire pour résoudre sa tâche.

Par la suite, le mécanisme d'attention est proposé par [Bahdanau et al., 2015]. Il est présenté plus en détail en section 2.1.2. Par exemple, dans le cadre de l'architecture *seq2seq*, ce mécanisme permet au décodeur d'accéder aux informations spécifiques à chaque élément encodé de la séquence en entrée au lieu d'avoir accès seulement à un unique vecteur qui compresse ces informations. On parle d'attention car au cours de l'entraînement, le modèle apprend sur quel élément il doit porter plus ou moins son attention à chaque étape du décodage. L'architecture `transformer` proposée par [Vaswani et al., 2017] est d'ailleurs une architecture *seq2seq* qui utilise ce mécanisme pour transférer les informations entre son encodeur et son décodeur. Ce mécanisme est alors aussi utilisé au sein de son encodeur, qui a la particularité d'être bidirectionnel afin que chaque élément de la séquence puisse avoir accès à l'ensemble des éléments de la séquence ; et au sein de son décodeur qui lui génère les éléments de gauche à droite.

Par la suite, l'architecture `transformer` est utilisé dans de nombreux travaux dans le but de pré-entraîner des modèles de langue sur de larges corpus de données. Parmi ces modèles, on peut citer notamment BERT [Devlin et al., 2019a], GPT [Radford et al., 2019] et T5 [Raffel et al., 2020]. Leur fonctionnement est expliqué en section 2.1.3. Ces modèles combinent un ensemble d'avancées réalisées au fur et à mesure des années en TAL, dont l'architecture *seq2seq*, le mécanisme d'attention mais aussi l'apprentissage auto-supervisé et l'apprentissage par transfert. Ces modèles profitent aussi de l'introduction de nouvelles tâches pour la modélisation du langage. Leur succès massif en TAL est dû à l'efficacité en termes de calcul de l'architecture `transformer`, aux larges corpus de données sur lesquels ils peuvent être entraînés de manière auto-supervisée, ce qui permet le transfert de connaissances acquises pendant le pré-entraînement vers une tâche cible. Ainsi des modèles multilingues reprenant l'architecture `transformer` ont aussi pu être entraînés comme mBERT, XLM, XLM-R et mT5. De plus, ces modèles de langue pré-entraînés sont rendus disponibles par leurs concepteurs, ce qui permet leur adoption dans de nombreux travaux, où ces modèles sont simplement fine-tunés sur une tâche cible ou incorporés dans des architectures plus complexes.

Ces dernières années, la majorité des travaux reprennent ces modèles de langue pré-entraînés. Certains travaux s'intéressent aux capacités de transfert zero-shot ou sur peu d'exemples (voir section 3.2.1) de ces modèles pré-entraînés, en utilisant ces modèles tels quels ou en les intégrants à une architecture spécifique. Une partie des travaux s'intéresse plus spécifiquement au format d'entrée à fournir à ces modèles comme le *prompting* [Liu et al., 2021a]. D'autres travaux s'intéressent à l'amélioration de l'architecture `transformer` et à de nouvelles approches afin d'adapter efficacement ces modèles à des tâches cibles, tout en maximisant le transfert [Houlsby et al., 2019].

2.1.2 Le mécanisme d'attention

Le mécanisme d'attention est au coeur de l'architecture `transformer` utilisée aujourd'hui par la majorité des modèles à l'état de l'art en TAL. Il est ainsi important d'en comprendre le fonctionnement. De plus, dans le chapitre 6, nous proposons une méthode de modulation de l'attention qui intervient directement sur ce mécanisme. Dans cette section nous décrivons ainsi le mécanisme d'attention en nous concentrant sur celui utilisé dans l'architecture `transformer`.

Idée générale

Dans [Vaswani et al., 2017], les auteurs expliquent « *Une fonction d'attention peut être décrite comme l'association d'une requête et d'un ensemble de paires clé-valeur à une sortie, où la requête, les clés, les valeurs et la sortie sont toutes des vecteurs. La sortie est calculée comme une somme pondérée des valeurs, où le poids associé à chaque valeur est calculé à partir une fonction de compatibilité entre la requête et la clé correspondante.* »[1]. Le mécanisme d'attention a ainsi pour but d'ajouter du contexte, formalisé comme une requête, à un état caché du modèle formalisé comme la paire clé-valeur. Ceci permet de créer des dépendances supplémentaires dans un modèle, puisque l'attention prend en entrée plusieurs vecteurs, ce qui peut être utile pour la résolution de certaines tâches. Un parallèle peut être fait avec les RNNs, qui permettent la prise en compte de la sortie précédente et qui sont ainsi particulièrement adaptés au traitement de séquences de mots par exemple, en particulier lorsqu'ils sont bi-directionnels.

Dans le cas de l'auto-attention (*self-attention* en anglais), la requête, la clé et la valeur correspondent à un même vecteur. Appliqué à une séquence de mots,

1. Traduit de l'anglais « *An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.* ».

ce mécanisme permet d'obtenir pour chaque mot une représentation vectorielle capable de prendre en compte le reste de la séquence, c'est-à-dire le contexte dans lequel le mot apparaît. Ceci est particulièrement utile pour les mots polysémiques comme le mot « bouton » en français, dont il est possible de comprendre le sens seulement avec du contexte. Par exemple, si deux phrases différentes contenant le mot « bouton » sont représentées à l'aide de plongements lexicaux comme word2vec [Mikolov et al., 2013], la représentation vectorielle de « bouton » dans ces deux phrases sera la même, même si ce mot n'a pas le même sens dans les deux phrases. Ainsi, l'auto-attention permet de prendre en compte le contexte de la phrase en entrée, et les deux représentations du mot « bouton » en sortie, issu de ces deux phrases, seront significativement différentes l'une de l'autre si le sens du mot est est différent dans les deux phrases.

Scaled-dot product

Plusieurs fonctions de « compatibilité » peuvent être utilisées. Nous décrivons ici l'attention utilisée dans l'architecture `transformer` [Vaswani et al., 2017], nommée « scaled-dot product ». Les entrées consistent en des requêtes et des clés de dimension d_k, ainsi que des valeurs de dimension d_v. Le produit scalaire de la requête avec chacune des clés est d'abord calculé et résulte en des scores. On parle de score car plus la requête est similaire à une clé, plus son score est important. Ces scores sont ensuite normalisés en les divisant par $\sqrt{d_k}$, puis une fonction softmax leur est appliquée pour obtenir des poids.

En pratique, l'attention peut être calculée sur plusieurs requêtes à la fois en rassemblant les requêtes dans une matrice Q. De la même manière, les clés et les valeurs sont rassemblées respectivement dans les matrices K et V. Ainsi, la matrice des sorties est calculée comme suit :

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{2.1}$$

Attention multi-tête

Le mécanisme d'attention multi-tête a été introduit par [Vaswani et al., 2017]. Il a pour but d'appliquer le mécanisme d'attention h fois en parallèle sur les mêmes requêtes, clés et valeurs. Chaque tête d'attention est indépendante et les sorties de chaque tête sont concaténées et transformées linéairement afin d'obtenir les mêmes dimensions que l'on aurait obtenues si un seul mécanisme d'attention était appliqué. Répéter l'attention sur plusieurs têtes permet ainsi au modèle se reposant sur ce mécanisme de se concentrer sur différentes parties et différents aspects de la

séquence en entrée. Par exemple, [Clark et al., 2019] montrent que certaines têtes d'attention de BERT [Devlin et al., 2019b] semblent être associées à des notions linguistiques spécifiques, comme les compléments d'objet direct d'un verbe ou les co-références.

L'attention multi-tête est formalisée comme suit :

$$h_t^s = MultiHead(Q, K, V) \tag{2.2}$$

$$= Concat(head_1, ..., head_h) \cdot W^O \tag{2.3}$$

$$\text{avec } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

Où h désigne le nombre de têtes d'attention et les matrices Q, K et V désignent respectivement les matrices de requêtes, clés et valeurs. Les projections (transformations linéaires) se font à l'aide des matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ et $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

2.1.3 Modèles de langue pré-entraînés

Comme indiqué précédemment, les modèles de langue pré-entraînés utilisés aujourd'hui sont tous issus de l'architecture `transformer`. Cette architecture repose notamment sur le mécanisme d'attention multi-tête présenté précédemment. Son encodeur et son décodeur sont en effet tous deux composés de 6 couches d'attention à 12 têtes qui sont superposées les unes par dessus les autres [Vaswani et al., 2017].

Ainsi, GPT [Radford et al., 2019] est construit à partir de la partie décodeur de l'architecture `transformer` et est entraîné de manière auto-supervisée avec comme objectif la prédiction du prochain mot, étant donné une phrase ou un texte en entrée. De son côté, BERT [Devlin et al., 2019a] est construit à partir de la partie encodeur de l'architecture `transformer`, qui a la particularité d'être birectionnelle, et est entraîné de manière auto-supervisée sur deux tâches : *Masked Language Modelling*, où le modèle doit prédire le mot masqué dans une séquence et où l'aspect bidirectionnel lui permet de considérer l'ensemble des mots de la séquence et pas seulement ceux situés à la gauche ; et la tâche de *Next Sentence Prediction* où le modèle doit prédire si les deux séquences présentées en entrée se suivent. Les deux tâches sont apprises conjointement dans le cadre de l'apprentissage multi-tâche. Après un pré-entraînement sur de larges corpus de données textuelles sur ces deux tâches, BERT peut ensuite être utilisé sur d'autres tâches cibles. Originellement, BERT était prévu pour être fine-tuné tel quel sur de nouvelles tâches, c'est-à-dire de reprendre l'entraînement du modèle, mais dans le cadre d'un entraînement supervisé sur une tâche précise (par exemple, la classification de texte, des tâches

de question-réponse ou des tâches d'annotation en séquence). Cependant, les sorties de BERT peuvent aussi être utilisées comme des représentations vectorielles contextuelles pre-entraînées pour représenter les séquences en entrée d'un modèle. Les poids de BERT peuvent alors être adaptés ou bien fixés pendant l'entraînement du modèle complet sur la tâche cible.

D'un autre côté, T5 [Raffel et al., 2020] repose sur l'architecture encodeur-décodeur `transformer` complète et est entraîné de manière multi-tâche sur un mélange de tâches auto-supervisées et supervisées toutes converties au format *text-to-text*. La tâche auto-supervisée consiste en une tâche de débruitage appelée *fill-in-the-blanks text generation* assez semblable à la tâche de *Masked Language Modelling* de BERT. En effet, avec T5 les seules différences sont qu'un segment de plusieurs mots peut être prédit et que les prédictions correspondent seulement aux mots manquants dans le texte en entrée. Les tâches supervisées correspondent de leur côté à des tâches de classification de texte, de résumé, de question-réponse et de traduction. Les concepteurs de T5 ont aussi rendu disponible une version entraînée seulement de manière auto-supervisée (T5v1.1).

2.2 Les systèmes de dialogue orientés tâche

2.2.1 Définitions et différences

Plusieurs types de systèmes de dialogue peuvent être identifiés [Deriu et al., 2020] : les systèmes de dialogue orientés tâche, les systèmes de dialogue conversationnels et les systèmes de Question-Réponse interactifs.

Systèmes de dialogue orientés tâche. Ces systèmes, aussi dits orientés but, ont pour objectif de réaliser certaines tâches et de récupérer des informations à la demande de l'utilisateur et ce dans un ou plusieurs domaine(s) spécifique(s), comme par exemple la réservation d'un billet d'avion ou la recherche d'un restaurant. Les interactions se font sur plusieurs tours de dialogue et l'utilisateur peut affiner ou modifier ses critères au fur et à mesure, voire changer d'objectif au cours du dialogue. Ainsi, le système doit être en mesure de garder en mémoire et de mettre à jour à chaque tour du dialogue les informations fournies par l'utilisateur et avoir une idée de son but courant. Ces systèmes reposent généralement sur une base de données associée au domaine (par exemple une base de données répertoriant les restaurants ainsi que leurs caractéristiques).

Systèmes de Question-Réponse interactifs. Ces systèmes se limitent à un format de dialogue simplifié qui consiste, comme le nom l'indique, en une question

de l'utilisateur suivie de la réponse du système. Ils sont dits « interactifs » lorsque le système permet à l'utilisateur de poser des questions complémentaires en lien avec la question initiale. Dans ce cas, le système doit savoir garder en mémoire les informations fournies au tour précédent pour gérer notamment les co-références. Par exemple, après avoir posé la question `"Quand est né Barack Obama ?"`, l'utilisateur peut demander `"Et il est né où ?"`. De tels systèmes reposent généralement sur une base de connaissances, une base de questions-réponses ou des documents textuels.

Système de dialogue conversationnels. Ces systèmes ne sont pas en mesure de réaliser de tâche et ne se rattachent pas à un domaine en particulier. Leur but est de permettre des interactions sociales les plus naturelles et les plus cohérentes possible avec leurs utilisateurs, d'où leur appellation de *social bot* ou de *chit chat system*. Afin d'être engageants et cohérents, de tels systèmes doivent garder en mémoire les informations qu'ils transmettent à l'utilisateur et que l'utilisateur leur transmet tout en ayant une connaissance du monde.

Autres systèmes. Les systèmes de dialogue que l'on peut rencontrer dans la vie de tous les jours sont communément appelés chatbots, cependant le terme de chatbot est générique et ne donne aucune indication sur le type exact du système. De plus, les systèmes de dialogue actuels tendent à permettre à leurs utilisateurs de réaliser des tâches spécifiques tout en étant ayant des interactions uniquement sociales. Les assistants personnels tels que Siri [2], Google Assistant [3] ou Amazon Alexa [4] en sont l'exemple. Ils sont en mesure de réaliser des tâches simples, comme régler une alarme ou de répondre à des questions précises tout en étant capables de raconter des blagues à la demande de l'utilisateur ou de répondre à une déclaration d'amour.

Dans le cadre de ce mémoire, nous nous intéressons uniquement aux systèmes de dialogue orientés tâche. La structure de ces systèmes ainsi que les tâches qui sont nécessaires à réaliser pour leur fonctionnement sont décrites dans la prochaine section. Pour chacune de ces tâches est aussi donné un état de l'art rapide des approches utilisées pour leur réalisation.

2.2.2 Tâches et état de l'art

Un système de dialogue orienté tâche est généralement séparé en plusieurs modules [Young, 2006]. Comme présenté en Figure 2.1 ces modules correspondent à :

2. https ://www.apple.com/es/siri/
3. https ://assistant.google.com/
4. https ://www.amazon.com/

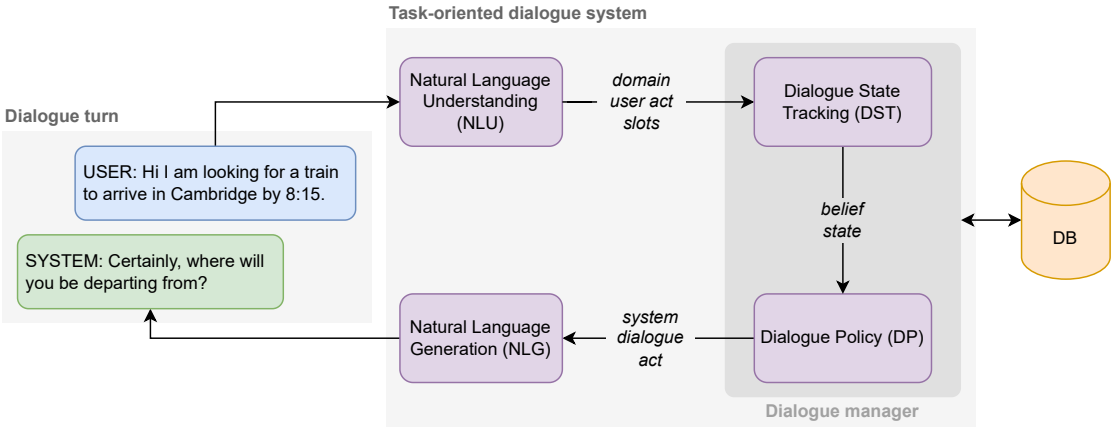


FIGURE 2.1 – Schéma général d'un système de dialogue orienté tâche à partir d'un exemple de tour de dialogue en anglais.

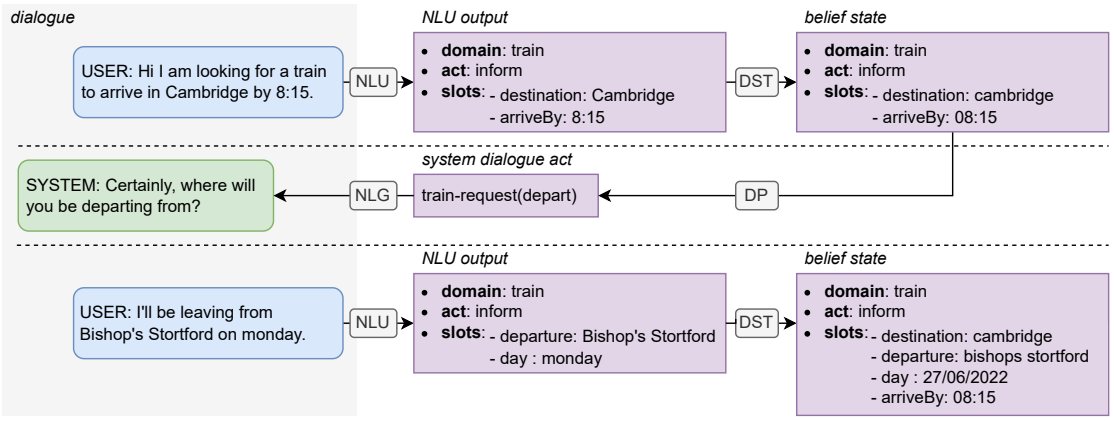


FIGURE 2.2 – Exemple de dialogue issu du corpus en anglais MultiWOZ [Eric et al., 2020]. Pour chaque énoncé utilisateur, sont données les sorties attendues de chaque module du système de dialogue afin de générer l'énoncé système adapté.

- La compréhension de la langue (NLU) : Identifie l'intention de l'utilisateur et extrait les informations associées, sous la forme de slots.
- Le suivi de l'état du dialogue (DST) : Met à jour les informations données par l'utilisateur à chaque tour.
- La politique de dialogue (DP) : Décide quelle(s) action(s) prendre en fonction de l'état du dialogue courant et des informations récupérées dans la base de données (DB).
- Génération de l'énoncé utilisateur en langage naturel à partir de l'acte de dialogue fourni par la politique de dialogue (NLG).

Un exemple de dialogue accompagné des sorties de chaque module est donné en Figure 2.2. Dans le cadre de ce mémoire, nous nous concentrons sur des entrées textuelles, mais dans le cas où les entrées sont vocales, celles-ci doivent d'abord passer par un module de reconnaissance automatique de la parole (*Automatic Speech Recognition* en anglais) et la sortie du module de NLG doit passer par un module de conversion de texte en parole (*Text-To-Speech* en anglais).

De nombreux travaux s'intéressent aussi à une approche bout-en-bout (*end-to-end* en anglais), où un unique modèle est entraîné à prédire directement les actes de dialogue du système, ou même l'énoncé du système, à partir de l'énoncé utilisateur en entrée [Bordes et al., 2017, Madotto et al., 2018]. Les travaux les plus récents tirent parti de modèles de langue pré-entraînés tels que GPT-2, BERT et T5 [Ham et al., 2020, Lee et al., 2021a, Lee, 2021]. Cette approche est intéressante car elle est moins coûteuse en termes de données d'entraînement et permet surtout d'éviter le problème de propagation des erreurs d'un module à l'autre. Cependant, elle rend les prédictions du modèles moins explicables et il est ainsi plus difficile pour les concepteurs d'un système de dialogue utilisant cette approche de corriger les comportements non désirés de leur système. En d'autres mots, une architecture modulaire permet un plus grand contrôle sur les sorties, ce qui explique qu'elle soit plus populaire auprès des industriels notamment.

De plus, il est important de noter que chaque système de dialogue ne suit pas forcément l'architecture modulaire présentée en Figure 2.1, puisque des modules supplémentaires peuvent être présents ou certains modules peuvent être fusionnés, comme par exemple les modules de NLU et de DST ou les modules de DST et de DP. L'architecture modulaire est présentée ici pour simplifier le discours.

Dans le reste de cette partie, nous décrivons les différentes tâches associées à chaque module ainsi que les modèles et approches permettant leur réalisation.

FIGURE 2.3 – Exemple d'un énoncé utilisateur dans le domaine des restaurants, annoté pour la tâche de détection des slots au format IOB.

Classification des actes de dialogue ou détection de l'intention.

Cette tâche est associée au module de NLU. Elle consiste à associer une étiquette à un énoncé utilisateur, l'étiquette décrivant la fonction de l'énoncé au sein du dialogue. Bien que la classification se fasse uniquement sur les énoncés de l'utilisateur, on peut aussi parler d'acte de dialogue du système. Généralement, on parle d'intention (*intent* en anglais) quand les étiquettes ont été définies par rapport au domaine d'application du système, comme dans le corpus ATIS où les intentions sont spécifiques au domaine de la réservation d'avion [Hemphill et al., 1990a]. Les différentes intentions peuvent aussi être très précises comme dans le cas du corpus Banking [Casanueva et al., 2020] qui contient 77 différentes intentions dans le domaine de la banque. Au contraire, les actes de dialogues sont en théorie définis de manière à être généraux et donc indépendants du domaine d'application [Mezza et al., 2018]. Cependant, lorsqu'ils sont appliqués aux systèmes de dialogue, les actes de dialogue à identifier sont restreints au minimum afin de faciliter leur détection. Dans le cas où les dialogues peuvent concerner différents domaines, les actes de dialogue peuvent aussi comprendre le nom du domaine afin de réaliser l'identification du domaine de manière jointe à la détection de l'acte de dialogue. Par exemple, dans le corpus MultiWOZ 2.1 [Eric et al., 2020], on pourra trouver les actes de dialogue `restaurant-request` et `general-greet` où la première partie indique le domaine et la deuxième partie l'acte de dialogue selon une nomenclature commune à tous les domaines. Cette tâche peut être apprise conjointement avec la tâche de détection des slots [Liu and Lane, 2016, Chen et al., 2019] ou séparément. En particulier, les approches s'intéressant à la détection des intentions en présence d'énoncés hors cadre (*out-of-scope*) ont tendance à ne pas considérer la tâche de détection des slots [Larson et al., 2019, Zhang et al., 2022]. Cette tâche est généralement évaluée à l'aide de la mesure d'*accuracy* accompagnée parfois de celle du F1-score.

Détection des slots.

Cette tâche est associée au module de NLU. Appelée *slot-filling* en anglais, elle consiste en la détection à chaque tour de parole des informations données par utilisateur qui sont nécessaires à la continuation du dialogue. Concrètement, à partir

de l'énoncé utilisateur courant, il s'agit de donner la liste des slots associés, un slot correspondant à une paire `(type du slot, valeur du slot)`. À noter que cette tâche se rapproche conceptuellement de la tâche plus générique de reconnaissance des entités nommées (*Named Entity Recognition* ou NER en anglais). Historiquement, il s'agissait de remplir un tableau, chaque colonne indiquant un slot, d'où le nom de *slot-filling*. Aujourd'hui, cette tâche est généralement traitée comme une tâche d'annotation de séquences, où une étiquette doit être associée à chaque token de l'énoncé utilisateur, suivant le format IOB. Si des slots sont identifiés dans l'énoncé, les tokens correspondant à la valeur du slot seront annotés avec l'étiquette `B-<type du slot>` pour le premier token (*Beginning*) et `I-<type du slot>` pour les autres tokens (*Inside*). Les autres tokens sont annotés avec l'étiquette `O` (*Outside*). Un exemple d'énoncé annoté est montré en Figure 2.3. Ces dernières années, la majorité des travaux se concentrent sur la détection conjointe de l'intention et des slots, les modèles à l'état de l'art sur les corpus traditionnels ATIS [Hemphill et al., 1990b] et SNIPS [Coucke et al., 2018] se reposant sur BERT [Devlin et al., 2019a], un encodeur transformer pré-entraîné [Chen et al., 2019, Qin et al., 2019]. Cependant, malgré des performances presque parfaites sur les corpus ATIS et SNIPS, ceux-ci offrent un cadre expérimental assez éloigné d'un cas réel. Ainsi, les travaux sur la détection des slots ont commencé à s'intéresser à des cadres expérimentaux plus complexes dont : l'intégration du contexte pour résoudre les ambiguïtés dans le cadre de dialogue sur plusieurs tours [Su et al., 2018] ; la détection des slots sur plusieurs domaines [Shah et al., 2019] ; et la détection des slots sur plusieurs langues avec des corpus comme MultiATIS++ [Xu et al., 2020a] ou le très récent corpus MASSIVE [FitzGerald et al., 2022] dont les baselines reposent sur les modèles multilingues pré-entraînés mBERT, XLM-R [Conneau et al., 2020a] et mT5 [Xue et al., 2021]. Pour plus d'informations, [Qin et al., 2021] fournit un état de l'art complet sur les tâches liées au NLU.

Suivi de l'état du dialogue.

Cette tâche consiste en la mise à jour de l'état du dialogue au fur et à mesure des interactions avec l'utilisateur. L'état du dialogue représente le but courant de l'utilisateur et prend la forme d'une structure sémantique reprenant l'intention générale de l'utilisateur ainsi que l'ensemble des critères qui accompagnent cette intention, qu'on appelle ici *slot* comme dans le cadre de la tâche de détection des slots. Le défi pour les modèles visant à réaliser cette tâche est de réussir à différencier les slots proposés par le système et les slots requis réellement par l'utilisateur et de gérer les co-références et les ambiguïtés, sachant qu'une valeur de slot peut avoir été énoncée seulement par le système et que certains éléments peuvent avoir été confirmés implicitement. Les données en entrée fournies au module DST dépendent de l'architecture du système de dialogue auquel est associé le module. Par

exemple, [Young, 2006] prend en entrées les sorties du module de NLU. Cependant, les travaux actuels ont tendance à travailler directement sur l'énoncé utilisateur. SUMBT [Lee et al., 2019] prend par exemple en entrée les énoncés d'un tour de parole, ainsi que le nom d'un slot, et prédit la valeur du slot en comparant les tokens mis en valeur par un processus d'attention aux valeurs de slot candidates. Le type du slot, les énoncés et les valeurs de slots sont encodés à l'aide de BERT et l'accès aux prédictions du tour précédent se fait grâce à un RNN. Le modèle à l'état de l'art sur le corpus MultiWOZ 2.1 [Eric et al., 2020] repose lui sur l'architecture TripPy [Heck et al., 2020] entraîné sur un curriculum d'exemples [Dai et al., 2021]. TripPy tire profit de trois mécanismes différents : il extrait les valeurs de slot directement des énoncés de l'utilisateur, il accède à une mémoire où sont répertorié les slots indiqués par le système à l'utilisateur, et il copie les slots déjà présents dans les états du dialogue du tour précédent. TripPy tire aussi profit de BERT [Devlin et al., 2019b] puisque chaque entrée est encodées par BERT. Récemment, l'attention s'est portée particulièrement sur le transfert entre domaines dans le cadre d'une approche zero-shot d'un nouveau domaine [Campagna et al., 2020, Li et al., 2021, Lin et al., 2021b] et sur le transfert d'une tâche de Question-Réponse à la tâche de DST [Lin et al., 2021a]. Nous conseillons [Jacqmin, 2022] pour un état de l'art plus complet.

Politique de décision.

La politique de décision permet au système de décider quelle action prendre au prochain tour étant donné l'état courant du dialogue. Dans sa forme la plus simple, la politique de décision est basée sur des programmes qui sont implémentés manuellement selon les indications d'experts du domaine. Dans ce cas, la mise à jour de l'état du dialogue se fait généralement aussi à l'aide de ces mêmes programmes [Brabra et al., 2022]. Ces programmes correspondent par exemple à des règles, qui vont associer à chaque état du dialogue une ou des actions à réaliser par le système. Ces règles peuvent se reposer sur un modèle définissant tous les états du système ainsi que toutes les transitions pouvant être identifiées. En pratique, il est impossible d'implémenter tous les cas de figure possibles, ce qui rend cette approche fastidieuse à mettre en place, peu flexible, difficile à maintenir et peu robuste notamment lorsque le nombre de tours de dialogue est important. Le dialogue peut aussi être schématisé sous la forme d'une structure sémantique décrivant toutes les informations nécessaires au système pour réaliser la tâche requise par l'utilisateur. La politique de dialogue consiste alors à demander à l'utilisateur les informations manquantes jusqu'à être en mesure de réaliser la tâche [Bobrow et al., 1977]. Bien que simple, cette approche est plus flexible que la précédente et est à la base d'assistants conversationnels comme Siri d'Apple, Alexa d'Amazon et l'assistant Google [Brabra et al., 2022]. Globalement, ces deux approches permettent de contrôler

avec précision les actions à prendre ainsi que les informations à communiquer ; ne nécessitent pas de données d'entraînement ; sont opérationnelles immédiatement ; et demandent peu de ressources de calcul, ce qui explique qu'elle sont encore populaires auprès des industriels. Cependant, ces approches sont fastidieuses à développer et à mettre à jour, en particulier lorsqu'il s'agit d'adapter la politique de dialogue à de nouveaux slots, intentions, actions ou domaines. Ainsi, en recherche, de nombreux travaux se sont intéressés dans un premier temps à des approches utilisant l'apprentissage par renforcement (RL, voir section 3.2.6) en définissant la fonction de récompense à partir de données, comme la réalisation de la tâche, la satisfaction de l'utilisateur, la durée du dialogue, etc. Le RL peut aussi être utilisé en ligne, ce qui permet d'avoir une politique de dialogue la plus cohérente possible avec les besoins des utilisateurs, en particulier si ceux-ci évoluent au cours du temps. Cependant, cette approche demande beaucoup d'interactions avec les utilisateurs avant d'être performante. L'apprentissage profond a ensuite été appliqué au RL permettant d'éviter la définition manuelle des caractéristiques sur lesquelles doit se baser l'algorithme de RL [Cuayáhuitl, 2017, Peng et al., 2017]. Néanmoins, cette approche demande beaucoup de données d'entraînement afin d'être efficace ou bien d'avoir recours à un utilisateur simulé pouvant présenter certains biais. D'autres travaux ont aussi entraîné directement des réseaux neuronaux notamment en considérant conjointement cette tâche avec la tâche de suivi de l'état du dialogue [Zhang et al., 2019a]. Cependant, ces approches sont moins performantes que celles reposant sur le RL puisqu'elles ne sont pas capables de considérer l'impact de l'action prédite par le modèle sur le succès du dialogue. Récemment, le RL et les réseaux à mémoire ont été conjointement appliqués afin de profiter de leurs avantages complémentaires [Zhao et al., 2021].

Génération de réponse (NLG).

Cette tâche consiste en la conversion d'un acte de dialogue du système en langage naturel. En Figure 2.2 par exemple, l'acte de dialogue du système `train-request (departure)` peut donner l'énoncé `"Certainly, where will you be departing from?"`. Dans sa forme la plus simple, cette tâche peut être réalisée à partir de patrons de phrases et de règles définis manuellement par des experts du domaine [Langkilde and Knight, 1998]. Cette tâche représente cependant un défi important pour l'apprentissage profond. Il est en effet primordial pour les systèmes de dialogue orientés tâche de contrôler avec précision les informations transmises à leurs utilisateurs, tout en offrant une interface engageante capable de s'exprimer de manière diversifiée. Ainsi, des architectures encoder-decoder augmentées d'un mécanisme d'attention ont été notamment proposées [Tran and Nguyen, 2017, Zhu et al., 2019]. Plus récemment, des travaux se sont intéressés à l'apprentissage de la tâche de NLG sur peu de données. [Peng et al., 2020] ont par exemple pré-entraîné

un modèle GPT-2 [Radford et al., 2019] sur un large corpus de données annotées de NLG. De leur côté, [Kale and Rastogi, 2020] tirent profit du modèle T5 [Raffel et al., 2020] (voir section 2.1.3), et lui fournissent en entrée soit l'acte de dialogue prédit par la politique de dialogue, auquel les noms des slots ont été remplacés par leur description, ou bien un énoncé basique généré à partir d'un patron spécifique à l'acte de dialogue, pour ensuite générer un énoncé correspondant à l'acte de dialogue. Cette approche est avantageuse puisqu'elle leur permet dans les deux cas d'appliquer leur modèle à de nouveaux slots et de nouveaux domaines. Cette tâche est généralement évaluée à l'aide des métriques BLEU [Papineni et al., 2002] et le *slot error rate*.

2.3 Discussion

Nous avons vu dans ce chapitre que la majorité des tâches liées au TAL, et notamment celles liées aux systèmes de dialogue orientés tâche, sont résolues grâce à l'apprentissage profond. Les nombreuses avancées ont été réalisées notamment grâce au pré-entraînement de modèles type `transformer` sur un large nombre de données non annotées et à leur libre mise à disposition afin de permettre un apprentissage par transfert sur des tâches spécifiques.

Cependant, comme indiqué par [Bender and Koller, 2020], il reste encore des progrès à réaliser en TAL et nous ne sommes pas encore au stade où nous pouvons affirmer que les machines « comprennent ». De plus, il est nécessaire de se méfier de l'effet Clever Hans [5], où un modèle peut se concentrer pendant l'apprentissage, sur des caractéristiques sur lesquelles il ne devrait pas. Par exemple, [Niven and Kao, 2019] ont montré que leur modèle basé sur BERT obtient de très bonnes performances pour la tâche d'identification de structures argumentatives dans un texte, et ceci en « exploitant des signaux statistiques fallacieux » comme la présence du mot `"not"`. Les auteurs nuancent cependant leurs propos en indiquant que leur modèle basé sur BERT semblait moins se reposer sur ces signaux qu'un BiLSTM par exemple. Cet effet est donc dangereux, car si les données d'évaluation sont construites de la même manière que les données d'entraînement, alors les performances du modèle pourront être élevées, alors qu'une évaluation sur les données construites différemment pourrait donner des performances très basses.

Ce phénomène montre l'importance du travail à réaliser sur les données comme suggéré par [Heinzerling, 2019] : à la fois au cours de leur collecte afin de proposer des données les plus diversifiées possibles et qui ne présentent pas de biais, en particulier faire en sorte que les données d'évaluation ne ressemblent pas en tous

5. https://towardsdatascience.com/deep-learning-meet-clever-hans-3576144dc5a9

points aux données d'entraînement ; et au cours de l'évaluation, où les données peuvent être modifiées - par exemple en modifiant l'ordre des mots ou en tronquant les phrases - afin de vérifier si le modèle se concentre sur des éléments qui font sens.

Il est aussi important que les données d'évaluation soient plus proches de la réalité en incorporant par exemple des données comprenant des mots jamais vus pendant l'entraînement ou des données que le modèle n'est pas en mesure de traiter, comme la présence de données du domaine mais auxquelles aucune classe ne peut être associée, de données hors domaine ou carrément hors contexte. En effet, comme vu en section 1.1 avec l'exemple d'un système de dialogue, les développeurs d'un système ne sont pas en mesure d'implémenter ou de permettre à leur système de gérer tous les cas de figure possibles, et de fait, après son déploiement, un système sera nécessairement confronté à ce type d'éléments inconnus. Il est alors nécessaire que les systèmes possèdent la capacité de traiter ces éléments inconnus de manière adéquate. Par exemple, dans le cadre d'un système de dialogue pour la réservation de restaurants, l'utilisateur peut être amené à demander l'adresse d'un restaurant qui n'a jamais été vu pendant l'entraînement mais qui figure dans la base de données ; ou donner des types de critères de recherche jamais vu pendant l'entraînement mais ajoutés à la base de données après coup, comme savoir si un restaurant possède une terrasse ; ou demander à réserver une chambre à l'hôtel associé au restaurant ; ou même adresser une requête tout à fait traitable par le système mais dans une langue sur lequel le modèle n'a pas été entraîné.

Dans la suite de ce mémoire, nous nous intéressons ainsi à la manière dont un système de dialogue peut gérer ce type de nouveaux éléments pendant l'inférence, et plus précisément à comment lui permettre de s'y adapter au cours du temps. Cette capacité s'éloigne du mode d'entraînement actuel, où un modèle est entraîné hors ligne puis évalué sur des données semblables aux données d'entraînement, mais correspond plus à ce qu'on attend en pratique des systèmes développés aujourd'hui comme les systèmes de dialogue orientés tâches. Cette capacité est appelée *Lifelong Learning* et est présentée dans le chapitre suivant.

Chapitre 3

Le Lifelong Learning

Dans ce chapitre, nous nous intéressons au Lifelong Learning (LL) et présentons son intérêt pour les systèmes de dialogue orientés tâche. En section 3.1, après avoir retracé l'historique de ce paradigme, nous commençons par donner notre définition du LL en présentant les cinq caractéristiques associées à notre définition, puis nous appliquons le LL aux systèmes de dialogue orientés tâche. Ensuite, en section 3.2 nous présentons plusieurs paradigmes en lien avec le LL et expliquons en quoi ils se différencient du LL. Puis en section 3.3, nous présentons deux systèmes opérationnels en lien avec le LL. Enfin, en section 3.4 nous discutons de l'intérêt du LL malgré sa complexité.

3.1 Définitions

3.1.1 Historique

Le Lifelong Learning (LL) a d'abord été pensé pour la robotique [Thrun and Mitchell, 1995]. La motivation derrière la définition de ce nouveau paradigme réside dans le constat fait à l'époque, qu'un robot a nécessairement des connaissances limitées du monde et que les tâches qu'il est en mesure de réaliser et les environnements dans lesquels il est en mesure d'évoluer sont, de fait, eux aussi limités. Les auteurs proposent alors de permettre à leur robot d'apprendre de manière autonome de ses propres expériences et de ses interactions avec son environnement tout au long de son utilisation. Ces expériences lui permettraient de corriger les connaissances qu'il avait *a priori* et d'en accumuler de nouvelles afin d'être en mesure de s'adapter à de nouveaux environnements ou de réaliser de nouvelles tâches au cours du temps. Les auteurs insistent en particulier sur la notion de transfert, indiquant qu'un robot réalisant du LL devrait être capable de tirer pro-

fit des connaissances acquises précédemment pour apprendre plus facilement de nouvelles tâches ou pour s'adapter plus facilement à un nouvel environnement.

Par la suite, le LL a été appliqué à l'apprentissage supervisé [Thrun, 1995]. L'idée était d'entraîner un modèle au cours du temps sur une séquence de tâches similaires. Les méthodes employées devaient permettre de bénéficier du transfert entre tâches et d'obtenir un modèle capable de généraliser dans le but de rendre l'apprentissage d'une nouvelle tâche plus efficace et moins coûteux.

Avec l'essor de l'apprentissage profond, le LL est étudié plus largement et prend aussi le nom d'apprentissage continu. Les travaux dans le domaine de l'apprentissage profond se concentrent alors majoritairement sur le problème de l'oubli catastrophique [Parisi et al., 2019] et cherchent à développer des techniques visant à le minimiser. En effet, au fur et à mesure de l'apprentissage de nouvelles tâches, de nombreux modèles observent une baisse importante de leurs performances sur les tâches apprises précédemment, d'où le terme d'oubli catastrophique.

Au cours des années, les travaux sur le LL auront ainsi mis de côté les caractéristiques suivantes dans la définition du LL : l'apprentissage du système en autonomie, au cours de son utilisation et en interaction avec son environnement, ainsi que l'étude du transfert en avant lors de l'apprentissage de nouvelles tâches comme présentés initialement par [Thrun and Mitchell, 1995].

Dans le cadre du projet « Learning to Interact with Humans by Lifelong Interaction with Humans » (LIHLITH), nous nous sommes interrogés sur la définition du LL et sur la manière dont il peut être appliqué aux systèmes de dialogue. En particulier, nous nous sommes intéressés aux aspects d'apprentissage autonome, en interaction avec son environnement et ayant lieu en production ; aspects généralement mis de côté dans les travaux récents sur le LL. Ces aspects sont aussi considérés par [Chen and Liu, 2016, Chen et al., 2018], même si l'accent reste porté sur l'apprentissage continu. Ainsi, dans [Veron, 2019], nous avançons que dans le cadre des systèmes de dialogue orientés tâche, le LL peut être appliqué à la fois aux modules du système qui reposent sur l'apprentissage profond, comme le module de NLU, et aux autres modules, tels que la base de données associée au système. Nous présentons aussi les trois étapes nécessaires pour la réalisation du LL :

(1) Détecter la présence d'un nouvel élément.
(2) Extraire et identifier le nouvel élément.
(3) Adapter le composant associé à ce nouvel élément.

Ces étapes sont d'ailleurs similaires à celles proposées dans [Thrun and Mitchell, 1995] : (1) expliquer, dans le sens de prédire l'état actuel de son environnement ; (2) analyser l'état de son environnement pour extraire les informations nécessaires à l'apprentissage ; (3) apprendre. Par la suite, la notion d'apprentissage sur le terrain

(*on-the-job learning*) a été introduite afin de caractériser ce type d'apprentissage qui se fait en production, en autonomie et en interaction avec les utilisateurs et son environnement [Liu, 2020], de manière à l'incorporer explicitement à la définition du LL. L'apprentissage sur le terrain reprend de fait les mêmes étapes que le LL.

Dans la section suivante, nous résumons les caractéristiques principales du LL tel que nous le définissons dans ce mémoire.

3.1.2 Caractéristiques

Le LL, tel que nous le considérons dans la suite de ce mémoire, décrit la capacité d'un système à être appliqué à et à apprendre plusieurs tâches au cours du temps, en production, en autonomie, en continu et de manière interactive. Ces différentes caractéristiques sont détaillées ci dessous. L'identification et la description de ces caractéristiques combinent les travaux réalisés avant la thèse [Veron et al., 2019, Veron, 2019] ainsi que les travaux d'autres chercheurs [Thrun and Mitchell, 1995, Chen et al., 2018, Liu, 2020].

Multi-tâche et au cours du temps. Le système doit pouvoir être appliqué à plusieurs tâches. Il doit de plus être en mesure de s'adapter à de nouvelles tâches au cours du temps. Il doit aussi être en mesure de conserver au maximum ses performances initiales sur les tâches qu'il aura apprises par le passé, en d'autres mots : minimiser l'**oubli**. Il doit aussi profiter des tâches apprises dans le passé pour faciliter l'apprentissage de nouvelles tâches à travers le **transfert** des connaissances acquises sur les tâches passées.

En production. L'adaptation du système doit se faire après que le système a été déployé, pendant qu'il est utilisé par ses utilisateurs finaux. L'environnement de production étant un monde ouvert, des éléments inconnus apparaîtront forcément au fur et à mesure des interactions avec les utilisateurs. Les données issues de ces interactions seront ainsi disponibles de manière non ordonnée. De fait, l'adaptation du système doit se faire à partir d'un flux continu de données pouvant contenir des nouveaux éléments. De plus, le système doit être adapté sans l'interruption du service.

En autonomie. Le système doit apprendre de manière proactive et auto-supervisée. Plus précisément, le système doit être en mesure de détecter quand un nouvel élément peut être appris, de récupérer et identifier les nouveaux éléments à apprendre et de s'adapter sans l'aide d'expert ou d'individu extérieur.

En continu. Une fois le nouvel élément identifié et jugé comme étant fiable (voir section 3.1.4), le système doit être en mesure de l'incorporer à son fonctionnement immédiatement. L'apprentissage doit donc se faire en continu et non de façon incrémentale.

Interactif. L'adaptation du système doit se faire grâce à ses interactions avec son environnement. Les réactions de l'environnement face aux actions du système pourront ainsi être analysées afin de déterminer si des nouveaux éléments peuvent être identifiés.

3.1.3 Lifelong Learning appliqué aux systèmes de dialogue

Si l'on considère les systèmes de dialogue, le LL doit par exemple permettre l'application et l'adaptation d'un tel système à de nouvelles valeurs de slots et de nouveaux types de slots dans le cas de la tâche de détection des slots (voir section 2.2.2). Ces nouveaux éléments peuvent être introduits par l'utilisateur ou via la mise à jour de la base de connaissances associée au système. Le système de dialogue doit de la même manière pouvoir être appliqué à de nouveaux environnements comme dans le cadre d'un nouveau domaine ou d'une nouvelle langue.

L'adaptation du système devra se faire en continu afin de se rapprocher le plus possible du processus d'apprentissage pouvant avoir lieu entre deux humains et afin de maximiser l'engagement [Yu et al., 2004].

L'adaptation d'un système de dialogue pourra se faire en interaction avec ses utilisateurs. Un système de dialogue apprenant tout au long de sa vie pourra ainsi profiter des retours de ses utilisateurs, par exemple suite à une erreur de compréhension. Le système pourra aussi de lui-même demander des informations à l'utilisateur. Idéalement, ces retours utilisateur devront se faire en langage naturel mais des réactions simples comme un indicateur binaire de satisfaction associé à chaque énoncé du système peut être accepté. À noter que les interactions avec les utilisateurs n'empêchent pas le caractère autonome de l'apprentissage puisque les utilisateurs n'ont, en théorie, aucune idée du fonctionnement interne du système de dialogue. Les interactions pourront aussi se faire avec des services externes comme la consultation de documents ou de pages internet [Komeili et al., 2022].

3.1.4 Considérations supplémentaires

Dans le cadre du LL, il est important de s'assurer de la fiabilité des informations collectées grâce aux interactions avec les utilisateurs ou du moins, de faire en sorte que les méthodes employées pour adapter le système à des nouvelles données soient robustes au bruit. En effet, il est possible que les techniques employées pour

extraire les nouvelles données soient source d'erreur, que l'utilisateur se trompe ou qu'il mette intentionnellement le système en erreur [Wolf et al., 2017]. Ceci implique la présence d'une unité de stockage spécifique pour ces données qui ne sont pas encore estimées fiables, ainsi que la présence d'un module supplémentaire capable d'estimer leur fiabilité, dans le cas où cette estimation n'est pas déjà incorporée dans un des modules existants.

Idéalement, un tel système devrait aussi être en mesure de s'adapter à chaque utilisateur en personnalisant ses interactions afin que les échanges soient plus efficaces et plus agréables. Il peut être cependant difficile d'estimer quelles données relèvent de la personnalisation et quelles données peuvent être généralisables. Il peut être aussi intéressant de prendre en compte la patience et les aptitudes de chaque utilisateur dans le cas où celui-ci est impliqué de manière active dans le processus d'apprentissage, c'est-à-dire si le système pose des questions à l'utilisateur afin d'acquérir de nouvelles connaissances [Li et al., 2017].

À terme, les interactions avec les utilisateurs pourront se faire de manière multimodale et pas seulement à partir de texte. Ainsi, le système pourrait bénéficier du transfert entre différentes modalités par exemple en récupérant des informations d'images. Des domaines de recherche comme la génération de légende d'images ou les systèmes de questions-réponses à l'aide d'images (*visual QA*) sont déjà largement étudiés et pourrait bénéficier au LL.

Le LL étant déjà complexe, ces considérations supplémentaires ne seront pas traitées dans le reste du mémoire mais il est important d'en avoir connaissance pour de futurs travaux.

3.2 Paradigmes associés au Lifelong Learning

De nombreux paradigmes déjà définis, notamment dans le domaine de l'apprentissage profond, sont liés au LL et leur étude permet indirectement l'étude du LL. Nous donnons ici une définition de ces paradigmes et décrivons des points qui les différencient du LL.

3.2.1 Apprentissage zero-shot et apprentissage sur peu d'exemples

L'apprentissage zero-shot correspond à l'application d'un modèle existant à une tâche cible, alors qu'aucune donnée de ce domaine cible n'aura été vue pendant l'entraînement. Pendant l'inférence sur la tâche cible, le modèle pourra disposer de données du domaine cible, par exemple sous la forme de base de données, de

documents textuels ou de description de la tâche [Zhao et al., 2022]. Ces données ne pourront cependant pas correspondre à des données annotées pour la tâche cible. L'apprentissage zero-shot repose sur le principe du transfert de connaissances d'une tâche à une autre, ce qui évite en principe la collecte et l'annotation de données de la tâche cible ainsi que l'entraînement d'un modèle spécifique à la tâche cible ou même le fine-tuning du modèle sur de nouvelles données. De manière générale on parle de tâche cible mais les cas de figure sont variés : il peut s'agir de l'application du modèle sur la même tâche et les même classes mais sur une nouvelle langue, sur la même tâche mais sur des classes différentes, sur la même tâche mais sur des domaines différents, ou sur une tâche différente.

L'apprentissage sur peu d'exemples (*few-shot learning* en anglais) reprend le même principe, cependant le modèle peut avoir accès à un faible nombre de données annotées pour la tâche cible. Le modèle peut alors être adapté sur ces données annotées et/ou les utiliser pendant l'inférence, comme récemment dans le cadre du *prompting* avec démonstration [Brown et al., 2020a, Gupta et al., 2022]. Cette capacité à apprendre à partir de peu de données se rapproche de la manière dont nous, humains, sommes capables d'apprendre : nous avons accumulé suffisamment de connaissances et d'expériences pour être en mesure d'apprendre à réaliser une tâche simplement à l'aide de quelques exemples.

L'apprentissage zero-shot et sur peu d'exemples permettent d'appliquer un même modèle sur plusieurs tâches tout en profitant du transfert, comme mis en avant dans LL, cependant il ne permet pas l'accumulation de connaissances via l'adaptation du modèle au cours du temps.

3.2.2 Apprentissage continu

L'apprentissage continu, ou *continual learning* en anglais, consiste en l'entraînement incrémental d'un même modèle sur une séquence, comme illustré en figure 3.1. On peut identifier trois types de scénarios d'apprentissage continu : l'apprentissage séquentiel de nouvelles classes (par exemple une nouvelle intention dans le cadre des systèmes de dialogue), dans ce cas la tâche et le domaine d'application restent les mêmes sur la séquence [Nilsback and Zisserman, 2008] ; l'apprentissage d'un groupe de classes relatives à un nouveau domaine (par exemple, détecter les intentions de l'utilisateur dans le domaine de la réservation de train puis dans la recherche de restaurant, puis sur d'autres domaines), où la tâche reste la même sur la séquence [Ke et al., 2021, Madotto et al., 2021] ; et l'apprentissage de nouvelles tâches [Sun et al., 2020]. À noter que jusqu'ici l'apprentissage séquentiel de nouvelles tâches se fait via la reformulation des différentes tâches en une tâche unique et plus générale afin que le modèle gère à chaque fois le même type d'entrée et de sortie. Ainsi en TAL, les travaux portant sur l'apprentissage continu de nouvelles

tâches s'attachent d'abord à modifier le format des données de chaque tâche de manière à suivre le format de la tâche de Question-Réponse (en anglais *Question-Answering*, QA) [McCann et al., 2018]. Par exemple, la tâche de traduction de l'anglais vers l'allemand peut se faire en considérant la question `"What is the translation from English to German?"`, en définissant le contexte comme la phrase en anglais à traduire et la réponse comme la phrase en allemand correspondant à la traduction de la phrase en anglais. Pour faciliter le discours, nous parlons par la suite uniquement de l'apprentissage de nouvelles tâches, mais employons le terme « tâche » de manière générale. L'idée de l'apprentissage continu est de permettre un apprentissage séquentiel dans le cas où les données de chaque tâche sont disponibles au cours du temps, tout en maximisant le transfert entre tâches et en ne nécessitant l'entraînement que d'un seul modèle. Deux types de transferts existent : le transfert en avant et le transfert en arrière. Le transfert en avant correspond au gain de performance de la tâche considérée, grâce aux tâches ayant été apprises avant la tâche considérée. Le transfert en arrière correspond lui au gain de performance de la tâche considérée, grâce aux tâches ayant été apprises après la tâche considérée. Dans la majeure partie des cas, le transfert en arrière est négatif : on parle alors d'oubli ou même d'oubli catastrophique lorsque celui-ci est important [McCloskey and Cohen, 1989]. La majorité des travaux portant sur l'apprentissage continu se concentrent ainsi sur le développement de méthodes et techniques visant à minimiser l'oubli [Parisi et al., 2019].

Ainsi, l'apprentissage continu est un entraînement multi-tâche qui se fait au cours du temps, de manière similaire au LL mais il se différencie du LL puisque l'apprentissage continu ne remplit pas les conditions d'apprentissage en production, en autonomie, en ligne et en interaction du LL puisque l'apprentissage se fait de manière incrémentale et puisque les données annotées sont disponibles directement à chaque étape.

3.2.3 Apprentissage en ligne

L'apprentissage en ligne (*online learning* en anglais) consiste en l'adaptation d'un modèle sur un flux continu de données. Ce mode d'apprentissage est pertinent lorsque les données évoluent rapidement et que le modèle doit être adapté en temps réel (par exemple pour la prédiction de la météo) ou lorsque les capacités de stockage sont limitées (par exemple dans le cas d'un système embarqué) et que les données sont disponibles petit à petit. Cet apprentissage est contraire à l'apprentissage par *batch*, aussi appelé apprentissage hors ligne, où toutes les données sont disponibles dès le début et où le modèle est entraîné en une fois. L'adaptation du modèle peut être continue ou incrémentale mais le stockage des données reste limité. Au cours du temps sont présentés au modèle de nouveaux exemples

d'entraînement, sachant que la tâche et le domaine restent dans la majeure partie des cas les mêmes. Ainsi, de manière similaire à l'apprentissage continu, un modèle apprenant en ligne peut être sujet au problème d'interférence catastrophique, où le modèle tend à oublier les données apprises précédemment au fur et à mesure de l'apprentissage de nouvelles données. Généralement cet apprentissage est appliqué à des cas où les données n'ont pas besoin d'être annotées (apprentissage non supervisé) ou sont annotées automatiquement sur des règles simples (apprentissage auto-supervisé). Dans le cas d'un système de recommandation d'articles sur internet, des données correspondent par exemple aux articles sur lesquels les utilisateurs cliquent. L'apprentissage en ligne a été principalement appliqué à des système d'apprentissage machine hors apprentissage profond. Comme les techniques employées sont très sensibles aux données d'apprentissage, les systèmes doivent souvent se munir d'un module permettant d'évaluer la qualité et la pertinence des données en entrée.

Récemment, l'apprentissage en ligne a été combiné à l'apprentissage continu afin de permettre l'apprentissage de nouvelles classes en ligne [Aljundi et al., 2019]. Cette configuration a notamment été appliquée au suivi et à la différenciation de visages dans des séries télévisées. Les nouvelles personnes pouvant apparaître au cours du temps sont considérées comme des nouvelles classes à apprendre. Dans le cas auto-supervisé le système suit l'hypothèse que si deux visages sont identifiés sur la même image, alors il s'agit nécessairement de deux personnes différentes. Pour ce faire, les auteurs ont modifié une méthode développée pour l'apprentissage continu dans le cadre de l'apprentissage profond, appelée *Memory Aware Synapse* [Aljundi et al., 2018], afin de l'adapter à un contexte d'apprentissage en ligne. Dans ce but, les auteurs ont défini un protocole pour « décider i) quand mettre à jour les poids d'importance, ii) quelles données utiliser pour les mettre à jours, et iii) comment accumuler les poids d'importance à chaque mise à jour ».

De manière similaire, [Coria et al., 2021] proposent un système permettant la segmentation et le regroupement en locuteurs (*speaker diarization* en anglais) en ligne capable de s'adapter à de nouveau locuteurs au cours du temps. Le système repose sur un modèle pré-entraîné capable d'associer une représentation vectorielle (*embedding* en anglais) à un flux audio en entrée et un algorithme de clustering pour associer la représentation vectorielle résultante à un locuteur précis (nommé arbitrairement). Leur méthode permet de gérer de nouveaux locuteurs grâce à une valeur de seuil telle que si la distance entre la représentation vectorielle en entrée et le centroïde le plus proche est supérieure à ce seuil, la représentation vectorielle est considérée comme appartenant à un nouveau locuteur. Les données servant au clustering sont ainsi augmentées au fur et à mesure de manière incrémentale.

L'apprentissage en ligne se différencie donc par rapport au LL puisqu'il n'implique

pas nécessairement l'adaptation sur de nouvelles classes et encore moins sur de nouveaux domaines ou nouvelles tâches. De plus, le transfert n'est pas considéré dans l'étude de ce paradigme.

3.2.4 Apprentissage en monde ouvert

L'apprentissage en monde ouvert (*open world learning* en anglais) est connu sous plusieurs noms, comme la détection de données hors distribution ou hors cadre (*out-of-distribution/scope detection*) ou la détection de nouveauté (*novelty detection* en anglais). Cependant, de manière générale, les travaux s'intéressent à la présence de données non attendues ou sur lesquelles le système n'a pas été défini initialement. Cette configuration s'oppose à l'hypothèse d'un monde fermé (*closed-world assumption* en anglais) sur laquelle reposent aujourd'hui beaucoup de travaux basées sur l'apprentissage profond. Ces travaux considèrent en effet que les données qui seront présentées pendant l'inférence pourront systématiquement être associées à des classes vues pendant l'entraînement. En production cependant, comme de nouveaux éléments apparaîtront inévitablement, il est nécessaire de détecter quand une entrée ne peut pas être traitée par le système, afin qu'il l'indique à son utilisateur au lieu de prédire une mauvaise classe.

Ainsi, [Hendrycks and Gimpel, 2017] reposent sur l'estimation de la confiance du modèle en ses propres prédictions afin de détecter les erreurs du modèle ainsi que les données hors distribution. Ils appliquent leur approche sur plusieurs tâches de TAL dont la classification de sentiments, la catégorisation de texte, l'étiquetage morpho-syntaxique et la reconnaissance automatique de la parole. Les auteurs présentent alors cette méthode comme une baseline, conscients que des améliorations sont possibles. [Hancock et al., 2019] montrent d'ailleurs que prédire que leur système de dialogue conversationnel a fait une erreur en reposant sur une mesure d'incertitude - à partir d'un seuil sur la probabilité maximale - est beaucoup moins performant qu'une approche reposant sur la classification de sentiments - en l'occurrence la prédiction de la satisfaction de l'utilisateur.

Dans le cadre des systèmes de dialogue orientés tâche, il est nécessaire de détecter les énoncés utilisateur ne pouvant pas être associés aux intentions que le système est en mesure de traiter, afin de ne pas entrer dans un processus de dialogue non adapté et de prévenir l'utilisateur. Ainsi, [Larson et al., 2019] ont proposé un premier corpus d'énoncés annotés en intention avec des données hors cadre, corpus nommé CLINIK [Zhang et al., 2021]. Par la suite, [Arora et al., 2020] ont aussi créé un corpus de données avec des énoncés hors cadre dont les données d'évaluation ont été collectées en production afin de représenter la réalité au plus proche. Les performances sur leur corpus étant bien plus basses que sur CLINIK, ceci montre l'importance de collecter des données en production. Ils ont de plus montré que

leur modèle peut avoir une confiance haute sur des énoncés hors cadre. De la même manière, [Vedula et al., 2022] s'intéressent à la tâche de détection des intentions en incorporant dans leur jeu de données d'entraînement des données hors domaine et en leur associant une étiquette unique `"classe inconnue"` ou leur propre label. Pendant l'évaluation, le système est évalué sur des données du domaine et des données hors domaine autres que celles présentées pendant l'entraînement. Si l'énoncé est annoté avec les classes des données hors domaine ou si le seuil de confiance est bas, alors l'énoncé est prédit comme étant hors domaine.

De leur côté, [Zhang et al., 2022] ont étudié l'impact de différents types d'intention hors cadre, sur les performances de différentes architectures `transformer` pour la détection des énoncés hors cadre. Partant du principe que la détection se fait à partir des scores de confiance du modèle sur ses prédictions, ils cherchent notamment à observer si les énoncés correspondant à différents types d'intentions hors cadre sont séparables des énoncés dont les intentions sont présentes dans les données d'entraînement à partir des scores de confiance. Ils observent ainsi qu'il est difficile de différencier les énoncés dont les intentions sont connues, des énoncés du domaine qui sont hors cadre, c'est-à-dire des intentions non présentes dans les données d'entraînement. À l'inverse il observe qu'il est plus facile de différencier les énoncés dont les intentions sont connues, des énoncés qui sont hors domaine et hors cadre.

L'apprentissage en monde ouvert s'intéresse ainsi seulement à la première étape nécessaire pour la réalisation du LL, étape qui correspond à détecter qu'un nouvel élément peut être appris.

3.2.5 Apprentissage sur le terrain

L'apprentissage sur le terrain a été introduit par [Liu, 2020]. Ce type d'apprentissage se concentre sur la capacité d'un système à apprendre après avoir été déployé, au cours de son utilisation. De fait, l'apprentissage doit se faire au cours du temps, en autonomie et en ligne. L'apprentissage peut aussi se faire grâce aux interactions du système avec ses utilisateurs. Cet apprentissage se fait en trois étapes comme présenté en section 3.1.1, étapes que nous ne restreignons pas à l'apprentissage machine de plusieurs classes contrairement à [Liu, 2020]. Ces étapes sont les suivantes : (1) Détecter la présence d'un nouvel élément, (2) Extraire et identifier le nouvel élément, et (3) Adapter les composants associés à ce nouvel élément. Comme indiqué par [Liu, 2020], la première étape est déjà étudiée dans le cadre de l'apprentissage en monde ouvert présenté en section 3.2.4 et la 3ème étape est étudiée à la fois dans le cadre de l'apprentissage continu présenté en section 3.2.2 et de l'apprentissage en ligne 3.2.3. Cependant, la 2ème étape a très peu été étudiée ainsi que la combinaison de ces 3 étapes.

L'apprentissage sur le terrain est assez proche de la définition de LL donnée précédemment mais elle est moins exigeante sur certains points. L'apprentissage sur le terrain peut être multi-tâche mais ce n'est pas une nécessité. Ainsi, un système de dialogue collectant au cours de ses interactions de nouveaux exemples d'entraînement afin d'améliorer son module de NLU et de coller au plus près des besoins de ses utilisateurs, ne sera pas caractérisé comme réalisant du LL puisqu'il n'apprend pas de nouvelle tâche ni même de nouvelle classe, mais comme capable d'apprendre sur le terrain. De plus, dans le cadre de l'apprentissage sur le terrain, nous ne nous cherchons pas à profiter du transfert afin de rendre les apprentissages futurs plus efficaces.

3.2.6 Apprentissage par renforcement

L'apprentissage par renforcement (en anglais, *Reinforcement Learning* ou *RL*) consiste en l'apprentissage d'une politique de décision permettant à un agent de décider quelle action prendre étant donné les observations de son environnement. L'apprentissage se fait grâce à l'accumulation d'expériences et grâce à une fonction de récompense permettant d'associer une récompense à chaque expérience. Les algorithmes de RL en apprentissage par *batch* (hors ligne) cherchent ainsi à trouver la politique de décision permettant de maximiser de cumul des récompenses sur l'ensemble des expériences disponibles pendant l'apprentissage. Le RL peut aussi être utilisé en ligne, de telle manière que la politique de décision est mise à jour à chaque interaction avec l'environnement. Des techniques récentes visent par ailleurs à combiner le RL hors ligne et en ligne afin de profiter des connaissances disponibles *a priori* tout en permettant au système d'être amélioré en ligne [Nair et al., 2021, Lee et al., 2021b]. Le RL a été appliqué avec succès à de nombreux domaines dont le TAL et plus spécifiquement aux systèmes de dialogue pour l'apprentissage de la politique de dialogue (voir section 2.2.2).

L'apprentissage par renforcement en ligne peut être une méthode pour aider l'amélioration du système au cours du temps mais il doit être combiné à d'autres méthodes afin de permettre l'apprentissage de nouvelles tâches sur le terrain et le transfert entre tâches.

3.2.7 Autres paradigmes

Apprentissage multi-tâche

Dans le cadre de l'apprentissage multi-tâche, un modèle apprend plusieurs tâches en simultané, sachant que certaines parties du modèle sont soit partagées entre les différentes tâches, soit spécifiques à une tâche mais liées d'une manière ou d'une autre. Ce type d'apprentissage est illustré en figure 3.1. Typiquement si un modèle

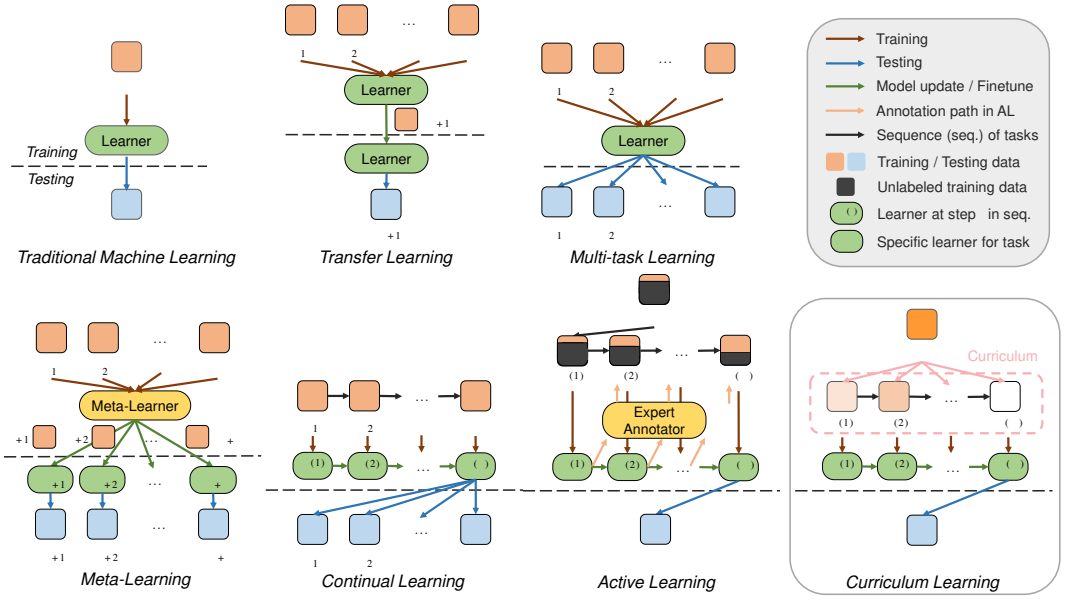


FIGURE 3.1 – Illustration de différents paradigmes d'apprentissage tiré de [Wang et al., 2022].

est amené à optimiser plusieurs fonctions de coût (*loss* en anglais) en simultané au cours de son entraînement, alors il s'agit d'un entraînement multi-tâche. Ce type d'apprentissage permet de transférer des connaissances entre tâches afin de généraliser et d'améliorer les performances du modèle sur chaque tâche tout en mettant à disposition un modèle capable d'être appliqué sur chacune de ces tâches. Plus d'informations sur l'application de ce paradigme à l'apprentissage profond peuvent être trouvées dans de récentes revues [Ruder, 2017, Crawshaw, 2020].

En TAL, on peut citer par exemple T5 [Raffel et al., 2020] qui est pré-entraîné sur une série de différentes tâches ou des modèles de NLU qui sont entraînés conjointement sur la tâche de détection des intentions et la tâche de détection des slots [Liu and Lane, 2016, Chen et al., 2019, Zhang et al., 2019b].

L'apprentissage multi-tâche se différencie du LL puisqu'il ne considère pas l'adaptation au cours du temps de son modèle sur de nouvelles tâches. En effet, toutes les données de toutes les tâches sont disponibles dès le début et un seul apprentissage est réalisé.

Meta-Learning

Le meta-learning consiste à entraîner un modèle sur plusieurs tâches permettant de généraliser afin que l'apprentissage de nouvelles tâches à partir de ce modèle soit

plus efficace et demande moins de données d'entraînement. Ce type d'apprentissage est illustré en figure 3.1. De fait, les algorithmes de meta-learning sont particulièrement adaptés à l'apprentissage zero-shot et l'apprentissage sur peu d'exemples (voir section 3.2.1). Ainsi, dans [Cattan et al., 2021], le meta-learning a été appliqué à la tâche de NLU. Les auteurs obtiennent ainsi un transfert entre langues plus important que lorsqu'un pré-apprentissage multi-tâche est réalisé sur les langues autres que la langue cible, autant dans le cas où peu d'exemples de la langue cible sont utilisés, que lorsque le modèle est appliqué directement à la langue cible (zero-shot). Nous encourageons le lecteur à consulter la revue [Lee et al., 2022] pour plus d'information sur le sujet.

Contrairement à l'apprentissage multi-tâche, le meta-learning est pensé pour être adapté à de nouvelles tâches. Cependant, dans le cadre du meta-learning, le modèle est adapté séparément sur chacune de ses tâches cibles, ce qui empêche le transfert entre tâches nouvellement apprises. Cet aspect différencie le meta-learning du LL.

Curriculum Learning

Introduit par [Bengio et al., 2009], le but du curriculum learning est que l'apprentissage du modèle commence par des exemples simples et que le niveau de difficulté des exemples augmente au fur et à mesure de l'apprentissage (voir illustration en figure 3.1). Cet apprentissage se rapproche de la manière dont nous, humains, apprenons : typiquement, nous cherchons d'abord à apprendre à différencier une tortue d'un chien, avant d'apprendre à différencier plusieurs espèces différentes de chiens. Cet apprentissage permet en théorie d'améliorer les capacités de généralisation du modèle et de rendre l'entraînement plus efficace, pouvant ainsi permettre d'obtenir de meilleures performances. Ce type d'apprentissage peut être appliqué à de nombreux modèles d'apprentissage machine [Wang et al., 2022]. Ainsi, [Dai et al., 2021] appliquent le curriculum learning à la tâche de DST (voir section 2.2.2) et au modèle TripPy [Heck et al., 2020], où des exemples de dialogues contenant notamment des co-références sont fournis au modèle pendant l'apprentissage après des exemples de dialogue plus simples. De cette manière, les performances sont améliorées de 5 points environ sur MultiWOZ 2.1 [Eric et al., 2020], ce qui permet au modèle d'obtenir les performances à l'état de l'art. Cependant, cette approche, bien que réalisant un entraînement progressif, suppose d'avoir à disposition toutes les données dès le début, ce qui la différencie particulièrement du LL.

Apprentissage avec un expert

L'apprentissage peut se faire de manière incrémentale avec l'intervention d'un expert à chaque incrément. Dans ce cas, on peut parler d'apprentissage avec l'humain dans la boucle (*human-in-the-loop* en anglais). Ce type d'apprentissage permet no-

tamment de limiter les exemples à annoter et d'accélérer le processus d'annotation de manière à diminuer les coûts liés à l'annotation. [Klie et al., 2021] utilisent par exemple le principe de pré-annotation pour la tâche d'appariement d'entités nommées, où un système recommande des segments de mots ainsi que des entités pour chaque entrée. Le système de recommandation est lui-même entraîné en continu et au cours du temps sur les annotations manuelles des annotateurs.

L'apprentissage actif, illustré dans la figure 3.1, est aussi un type d'apprentissage nécessitant la présence d'un expert. Au cours de l'apprentissage, le modèle a accès à des données non annotées et sélectionne dynamiquement les données pour lesquels il serait nécessaire d'obtenir des annotations. Le modèle peut par exemple sélectionner des données sur lequel il est peu confiant. Après avoir fait annoter ces données par un expert, le modèle est entraîné sur l'ensemble des données qui ont été annotées depuis le début. Le processus est répété jusqu'à atteindre un critère défini (par exemple l'atteinte d'un palier de performance sur le jeu de données de développement).

Apprendre des données de production

Lorsqu'un modèle est entraîné hors ligne sur des données puis mis en production, il arrive souvent de constater que les données d'entraînement n'étaient pas assez complètes ou qu'elles ne reflétaient pas exactement la réalité. Ainsi, certains travaux s'intéressent à la collecte de données de production et à l'adaptation hors ligne de leur modèle sur ces nouvelles données. Généralement, les approches utilisées pour l'adaptation reposent sur le transfert ainsi qu'un apprentissage semi-supervisé, ce qui leur permet d'utiliser un large de nombre de données de production sans avoir à les annoter.

Ainsi, [Hancock et al., 2019] ont cherché à améliorer leur système de dialogue conversationnel à partir de données de production. Leur système est composé de trois modèles associés aux tâches `DIALOGUE`, `SATISFACTION` et `FEEDBACK` ayant respectivement pour but de prédire l'énoncé du système, la satisfaction de l'utilisateur et l'énoncé de l'utilisateur. Les trois modèles sont entraînés hors ligne, sachant que les modèles pour `DIALOGUE` et `FEEDBACK` sont entraînés conjointement. En production, le système collecte de nouveaux exemples d'entraînement grâce à la prédiction de la satisfaction et ré-entraîne hors ligne et de manière périodique les modèles `DIALOGUE` et `FEEDBACK` sur l'ensemble des données disponibles.

De leur côté, [Xu et al., 2022] s'intéressent de plus près au type de retour pouvant être fait par l'utilisateur en cas d'erreur de la part du système et comparent différents algorithmes pour apprendre de ses retours. Leur système est d'abord entraîné sur des données initiales ainsi que des données collectées en production, puis il est

adapté sur un autre jeu de données collecté lui aussi en production mais plus tard, chacun des entraînements ayant lieu hors ligne.

Ces travaux sont très intéressants d'un point de vue du LL puisqu'ils contiennent presque toutes les caractéristiques du LL sauf celle de l'adaptation en ligne. De plus, comme il s'agit de systèmes de dialogue conversationnels dont le but est de mener une conversation sur n'importe quel sujet, il est difficile d'estimer l'aspect multi-tâche de leur apprentissage. Cependant, on peut considérer par extension que cette caractéristique est validée, puisqu'au cours du temps le système est en mesure d'accumuler des connaissances sur le monde qui améliorent ses interactions futures ainsi que les futurs apprentissages.

3.3 Systèmes liés au Lifelong Learning

Plusieurs systèmes en lien avec le LL ont été créés. La majorité de ces systèmes sont présentés dans d'autres sections, cependant certains n'ont pas pu être intégrés au contenu de ces sections. Nous présentons ainsi deux de ces systèmes dans cette partie, sachant qu'ils nous intéressent particulièrement car ils consistent en des systèmes opérationnels.

En 2010, [Carlson et al., 2010] ont proposé un système nommé NELL (« never-ending language learner ») dont le but est d'agrandir sa base de connaissances au cours du temps de manière infinie, tout en améliorant ses compétences. La base de connaissances est constituée de faits représentés généralement par deux entités et un prédicat (e.g. `playsInstrument(George_Harrison, guitar)`). Pour apprendre au cours du temps, le système réalise chaque jour ces trois étapes : (1) il essaie d'extraire de nouveaux faits à partir des pages internet qu'il consulte, (2) il essaie de déduire de nouveaux faits à partir des liens qu'il peut découvrir entre les faits de sa base de connaissances et supprime les faits devenus incorrects, et (3) il essaie d'améliorer ses capacités d'extraction et de déduction afin de réaliser les deux étapes précédentes de manière plus efficace et plus exacte, notamment en ré-entraînant ses modèles. Notez qu'un score de confiance est associé à chaque fait de la base. Entre 2010 et 2018, NELL a ainsi collecté plus de deux millions de faits considérés comme hautement fiables [Mitchell et al., 2018]. Il serait intéressant de combiner le système NELL à un système de dialogue afin de tirer profit de la base de connaissances sans cesse améliorée de NELL et des retours des utilisateurs sur ces connaissances. De plus, l'utilisateur pourrait être intégré dans le processus d'apprentissage en aidant le système à découvrir de nouveaux prédicats et à créer de nouveaux faits de manière similaire au travail de [Mazumder et al., 2019b].

D'un autre côté en 2020, le système de dialogue conversationnel Blender Bot a

été mis à disposition en *open source* [Roller et al., 2021]. Ce système permettait d'échanger avec un utilisateur dans le cadre de conversations ouvertes et était le premier capable de combiner des compétences conversationnelles comme la gestion de personnalité, le fait de faire preuve d'empathie et le fait de tirer profit de ses connaissances lors de ses interactions. Par la suite, le système a été amélioré notamment en y intégrant la possibilité d'accéder au web durant ses conversations avec ses utilisateurs afin intégrer des connaissances qui sont en phase avec l'état réel du monde [Komeili et al., 2022]. En effet, la majeure partie des systèmes de dialogue conversationnels actuels reposent sur des connaissances sauvegardées dans les poids de leur modèle lors de l'entraînement de celui-ci. Étant donné que le monde évolue, ces connaissances peuvent alors devenir incorrectes au cours du temps. Récemment, le système a été amélioré une troisième fois [Shuster et al., 2022] et permet notamment l'apprentissage à partir de données de production en intégrant notamment le travail de [Xu et al., 2022] présenté en section 3.2.7.

3.4 Discussion

Nous avons vu dans ce chapitre que le LL est complexe, et qu'il est lié à plusieurs paradigmes d'apprentissage qui constituent individuellement encore aujourd'hui des sujets de recherche importants. On peut ainsi se demander pourquoi il est pertinent de s'intéresser à un tel sujet. Nous estimons que le LL doit être vu comme un objectif, qu'il est important de considérer pour juger de la pertinence de l'étude de certains sujets ou à l'inverse pour constater le manque de recherche sur d'autres sujets. Ainsi, le LL rappelle les motivations derrière nombre de paradigmes comme l'apprentissage zero-shot ou sur peu d'exemples, l'apprentissage en ligne ou l'apprentissage continu. Le LL met aussi en lumière le manque de recherche sur l'apprentissage sur le terrain malgré son intérêt évident en pratique. Dans le cas précis de l'apprentissage continu, le LL permet de remarquer que le développement de méthodes ayant pour but de favoriser le transfert en avant est beaucoup moins mis en avant que le développement de méthodes visant à minimiser l'oubli.

Ainsi, dans le reste de ce mémoire, nous nous intéressons à deux aspects principaux du LL qui sont l'apprentissage sur le terrain et le transfert. L'ordre des chapitres repose sur la tâche étudiée et le type des nouveaux éléments sur lesquels le modèle considéré doit s'adapter en termes de difficulté. Ainsi, dans le chapitre 4, nous nous intéressons d'abord à la tâche de détection des slots et à l'apprentissage sur le terrain de nouveaux patrons de phrases et de nouvelles valeurs de slot. Ensuite, dans le chapitre 5, nous étudions le transfert entre langues dans le cadre de la tâche de détection des slots et de l'apprentissage continu sur une séquence de langues. Enfin, dans le chapitre 6, nous étudions le transfert zero-shot entre domaines dans

le cadre de la tâche de suivi de l'état du dialogue, où le modèle doit être en mesure de prédire des types de slots jamais rencontrés pendant l'entraînement ; et nous proposons des solutions afin d'améliorer le transfert.

Chapitre 4

Application et évaluation de l'apprentissage sur le terrain

4.1 Introduction

Dans ce chapitre, nous définissons une méthodologie générale pour évaluer les systèmes de dialogue apprenant sur le terrain. Nous décrivons ensuite un système de dialogue orienté tâche capable d'apprendre sur le terrain et nous l'évaluons selon la méthodologie définie au préalable. En effet, des systèmes de dialogue apprenant sur le terrain ont été construits et évalués ces dernières années, mais aucune méthodologie d'évaluation générale n'a été définie. Nous nous concentrons ici uniquement sur l'apprentissage sur le terrain, un apprentissage qui se fait en ligne, au cours du temps et en autonomie dans un milieu de production qui tire profit des interactions du système avec ses utilisateurs. Nous ne nous intéressons donc pas ici à l'aspect multi-tâche du LL, et, de fait, les systèmes et la méthodologie d'évaluation que nous décrivons ne considèrent pas l'apprentissage de nouvelles classes ou de nouvelles tâches. En effet, comme il n'existe à notre connaissance aucun système de dialogue capable d'apprendre selon l'ensemble des critères du LL, nous avons décidé de nous concentrer uniquement sur l'évaluation de l'apprentissage sur le terrain, afin de bénéficier de points de comparaison et d'appliquer notre méthodologie à un système. De plus, peu de travaux se sont intéressés à l'amélioration de la compréhension de la langue (NLU), bien qu'il s'agisse d'une des premières étapes nécessaire au bon fonctionnement d'un système de dialogue orienté tâche. Ainsi, nous décrivons un système de dialogue orienté tâche, capable d'améliorer sur le terrain ses performances sur la tâche de détection des slots. Développer notre propre système apprenant sur le terrain nous permet d'appliquer la méthodologie d'évaluation définie au préalable, mais aussi de se rendre compte

des difficultés et des points importants à considérer lors de l'étude de l'apprentissage sur le terrain et de son évaluation, ainsi que des limites de la méthodologie d'évaluation proposée.

Pour résumer, voici les contributions qui sont décrites dans ce chapitre :

- Une première définition d'une méthodologie d'évaluation générale pour les systèmes de dialogue apprenant sur le terrain.
- Un système de dialogue orienté tâche qui améliore sur le terrain son composant de NLU via la collection et l'inférence de nouveaux exemples d'entraînement grâce à ses interactions avec ses utilisateurs.
- L'évaluation du composant de compréhension de la langue du système précédent selon la méthodologie définie avec simulation de l'utilisateur et de la complétion de la base de connaissances associée au système.

Ces contributions ont été présentées au workshop NeurIPS HLDS (Human-in-the-loop dialogue systems) de 2020 [Veron et al., 2020].

Dans un premier temps, nous présentons en section 4.2 plusieurs systèmes de dialogue en lien avec l'apprentissage sur le terrain et décrivons la manière dont ils sont évalués, puis nous présentons des méthodologies d'évaluation liées à l'apprentissage sur le terrain. Ensuite, en section 4.3 nous présentons notre méthodologie générale pour l'évaluation de l'apprentissage sur le terrain d'un système de dialogue. De manière à tester notre méthodologie d'évaluation, nous présentons en section 4.4 le système de dialogue que nous avons implémenté, puis nous décrivons en section 4.5 la configuration de nos expériences. Enfin, en section 4.6 nous donnons les résultats de nos expériences sur notre système de dialogue dont l'apprentissage sur le terrain est évalué à l'aide de la méthodologie définie au préalable. Nous discutons aussi les résultats et la méthodologie d'évaluation.

4.2 État de l'art

Comme décrit en section 3.2.5, trois étapes sont nécessaires pour permettre l'apprentissage sur le terrain : (1) détecter quand un nouvel élément peut être appris, (2) récupérer et identifier le nouvel élément, et (3) adapter le(s) composant(s) associé(s) au nouvel élément. Dans la première partie de cette section, nous présentons comment ces trois étapes sont réalisées par différents systèmes de dialogue apprenant sur le terrain. Cette partie illustre notamment le fait que le type de nouveaux éléments à apprendre et les méthodes qui peuvent être utilisées pour réaliser ces trois étapes dépendent principalement du composant à améliorer. Pour chaque système nous présentons aussi la manière dont il est évalué. Dans une seconde partie, nous nous intéressons aux méthodologies d'évaluation utilisées dans le cadre d'autres types d'apprentissage et afin d'identifier les points intéressants,

ou à l'inverse les points manquants, pour l'évaluation de l'apprentissage sur le terrain.

Systèmes de dialogue apprenant sur le terrain

[Mazumder et al., 2019a] se sont intéressés à l'amélioration du composant de NLU sur le terrain. Ils ont construit un système qui associe des requêtes utilisateur formulées en langage naturel à des patrons de phrases décrivant une action spécifique (par exemple `"draw a X1 circle at X2"`) afin de réaliser par la suite cette action. Leur approche consiste alors à apprendre sur le terrain des nouveaux patrons de phrases. Pour détecter si un nouvel élément peut être appris, le système demande explicitement à son utilisateur si sa requête a été correctement comprise. Dans le cas où une erreur de compréhension a été commise, l'utilisateur doit explicitement indiquer quelle était l'action qui devait être associée à sa requête. Ainsi, le système est en mesure d'extraire un nouveau patron de phrase à partir de la requête initiale de l'utilisateur, de telle manière qu'une nouvelle manière d'exprimer l'action associée puisse être ajoutée au système. Le processus d'apprentissage est continu puisque l'adaptation du système est immédiate. Cependant, la liste des actions étant fixe, l'apprentissage se fait sous l'hypothèse de monde fermé (*closed world assumption* en anglais) et le système n'est pas en mesure d'apprendre de nouvelles actions au cours du temps. Des requêtes utilisateur en langage naturel ont été collectées et annotées avec l'action correspondante afin d'évaluer les différentes versions de leur système, celles-ci incluant une version où le système n'a pas été amélioré sur le terrain. Cependant, ces requêtes servent à la fois pour simuler la période de production où le système peut apprendre sur le terrain et pour évaluer l'apprentissage sur le terrain. Ainsi, leur évaluation permet de rendre compte d'un apprentissage sur les données présentées en production mais pas de rendre compte d'un apprentissage général pouvant être appliqué à d'autres données ou d'autres cas de figure. De plus, leur évaluation ne permet pas d'observer l'évolution des performances de leurs variantes au cours de l'apprentissage sur le terrain.

La base de connaissances associée à un système de dialogue (*Knowledge Base* en anglais, abrégé KB) peut aussi être améliorée sur le terrain, comme proposé par [Mazumder et al., 2019b]. Leur approche s'applique à un système de question-réponse interactif dont le but est de répondre à l'aide de sa KB à des questions factuelles d'un utilisateur. Leur approche permet d'enrichir la KB avec de nouveaux faits, grâce aux interactions du système avec ses utilisateurs et grâce à un système d'inférence. Nous considérons ici un fait comme un triplet composé de deux entités et d'un prédicat, tel que le triplet `(Obama, BornIn, Hawaii)` est composé des entités `Obama` et `Hawaii` et du prédicat `BornIn`. Quand un utilisateur pose une question qui se réfère à des éléments inconnus dans la KB (prédicat ou entité), le

système peut demander à l'utilisateur un exemple de fait contenant l'élément inconnu. Ce nouveau fait peut permettre au système d'inférer d'autres nouveaux faits qui peuvent ensuite rendre possible au système de répondre à la question initiale de l'utilisateur. Cette manière d'améliorer la KB à travers ce processus d'inférence est très proche de la définition de LL. Cependant, ces travaux se concentrent uniquement sur le moteur de recherche et d'inférence de fait et travaillent directement avec des requêtes sous la forme de triplet (par exemple `(Obama, BornIn, ?)`) et non pas des requêtes en langage naturel, ce qui simplifie grandement les étapes de détection et de récupération et identification des nouveaux éléments à apprendre. Pour simuler l'environnement de production, les auteurs ont utilisé un utilisateur simulé qui peut poser des questions avec des entités ou des prédicats inconnus, et donner des exemples de faits comprenant ces éléments inconnus. Ensuite, ils ont évalué leur système à la fin de l'ensemble des interactions avec l'utilisateur simulé sur les mêmes requêtes que celles données par l'utilisateur simulé. Ainsi, le processus d'évaluation n'est pas continu et ne permet pas de comparer l'apprentissage sur le terrain de versions de leur système au cours du temps.

D'un autre côté, [Hancock et al., 2019] ont construit un système de dialogue conversationnel qui collecte des nouveaux dialogues au cours de ses conversations avec ses utilisateurs. Ces nouveaux dialogues sont utilisés comme nouveaux exemples d'entraînement pour adapter périodiquement les modèles associés au système. Leur approche repose sur la prédiction de la satisfaction utilisateur après chaque réponse du système. Si l'utilisateur semble satisfait, le système stocke le dialogue comme un nouvel exemple d'entraînement (principe d'imitation). Sinon le système demande à l'utilisateur de lui donner une réponse qui aurait été adéquate et la réponse du système est remplacée dans le dialogue par celle proposée par l'utilisateur, afin que ce dialogue soit utilisé comme exemple d'entraînement. Pour permettre au système d'apprendre sur le terrain, différentes versions du système ont été déployées et testées sur une plateforme de crowdsourcing. Ensuite, ces versions ont été évaluées sur un jeu de données d'évaluation séparé à la fin du processus d'apprentissage sur le terrain. Ainsi, le processus d'évaluation n'est pas continu et ne permet pas de comparer l'apprentissage sur le terrain de versions de leur système au cours du temps.

Méthodologies d'évaluation

[Chen and Liu, 2016] ont décrit une méthodologie d'évaluation pour le LL adaptée à l'apprentissage automatique. Cette méthodologie consiste à comparer les performances de différents algorithmes sur une nouvelle tâche T_{n+1}. Plus précisément, il s'agit de comparer des algorithmes de LL qui apprennent la tâche T_{n+1} après avoir appris de manière séquentielle les tâches $\{T_i\}_{i\in[\![0,n]\!]}$, à des algorithmes

d'apprentissage en isolation qui n'apprennent que la tâche T_{n+1}. Cependant, cette méthodologie permet seulement d'évaluer l'aspect de transfert d'un ensemble de tâches vers une nouvelle tâche liée au LL tel que défini en section 3.1.2. Ainsi, elle ne permet pas une évaluation en continu, et encore moins l'évaluation des capacités d'un système à apprendre en autonomie dans un milieu de production.

Comme présenté en section 3.2.2, l'apprentissage séquentiel de différentes classes ou différentes tâches est aussi nommé apprentissage continu. Les travaux sur ce sujet évaluent généralement leur approche de manière incrémentale puisque l'apprentissage en lui-même se fait hors ligne, tâche par tâche et donc de manière incrémentale. Les métriques utilisées correspondent généralement à des mesures du transfert en avant ou en arrière, aussi appelé oubli catastrophique et à une mesure de la moyenne des performances sur chaque tâche. Les métriques sont calculées à la fin de la séquence d'apprentissage, ou moyennées sur l'ensemble des étapes de la séquence. Elles peuvent aussi être tracées au cours du temps, mais l'unité de temps correspond simplement à l'apprentissage hors ligne d'une nouvelle tâche. L'unité de temps reste aussi la même dans le cadre des travaux plus récents étudiant l'apprentissage continu en ligne [Aljundi et al., 2019]. Lorsque les métriques sont tracées au cours du temps, il est ainsi possible de comparer les différentes approches et observer comment leurs performances évoluent au fur et à mesure de l'apprentissage de nouvelles tâches, ce qui peut permettre d'observer des comportements non observables lorsque l'on se contente de regarder des performances moyennes ou finales.

Les méthodologies d'évaluation existantes qui seraient intéressantes pour l'évaluation de l'apprentissage sur le terrain sont celles utilisées dans le cadre de l'apprentissage par renforcement en ligne (voir section 3.2.6). L'évaluation peut se faire dans un environnement virtuel simulé afin de permettre l'évaluation dans les mêmes conditions dans le cadre de travaux différents [Todorov et al., 2012]. L'évaluation est réalisée sur plusieurs épisodes, un épisode consistant en un ensemble d'interactions au bout desquelles le système revient à son état initial. À la fin de chaque épisode, la moyenne des récompenses cumulées (*average return* en anglais) ou le taux moyen de succès sur l'ensemble des épisodes ayant eu lieu jusqu'ici sont calculés [Nair et al., 2021, Lee et al., 2021b]. Ce type de méthodologie permet de comparer différentes approches au cours du temps afin de voir si une approche apprend plus vite qu'une autre ou si certaines atteignent des plateaux d'apprentissage par exemple.

Ces deux dernières méthodologies sont intéressantes pour l'évaluation au cours du temps et la possibilité de comparer les différentes approches qu'elles proposent. Cependant, elle ne permettent pas l'évaluation des capacités d'un système à apprendre en autonomie dans un milieu de production.

Positionnement des travaux

Puisque qu'il n'existe pas de consensus sur la manière d'évaluer l'apprentissage sur le terrain pour les systèmes de dialogue, il est nécessaire de définir une méthodologie d'évaluation adaptée. Ainsi, nous décrivons en section 4.3 une première définition de méthodologie d'évaluation qui prend en compte l'aspect continu de l'apprentissage et qui clarifie la différence entre les données utilisées pour les phases de production et d'évaluation. Puisque l'amélioration sur le terrain de la tâche de détection des slots n'a jamais été étudiée pour à notre connaissance, nous décrivons ensuite en section 4.4 le fonctionnement d'un système de dialogue orienté tâche qui améliore sur le terrain ses performances sur la tâche de détection des slots dans le domaine de la cuisine. Ce système nous permet de plus d'éprouver la méthodologie d'évaluation et d'observer les avantages et les limitations de cette méthodologie comme présenté en section 4.6.

4.3 Méthodologie d'évaluation

Nous considérons ici un système de dialogue capable d'adapter de manière continue et en autonomie un de ses composants sur une tâche spécifique au cours de ses interactions avec ses utilisateurs en production (apprentissage sur le terrain). La tâche en elle-même reste la même au cours du temps mais le cadre, voire même le domaine d'application pourrait évoluer. Le composant a accès dans un premier temps à des données initiales, résultant en un état initial du composant associé à des performances initiales. Le système est ensuite exposé à des situations et des éléments inconnus lors de son utilisation en production (environnement ouvert) et doit adapter son composant à ces nouveautés au cours du temps, résultant en différents états de son composant au cours du temps. Un schéma récapitulatif de l'amélioration sur le terrain d'un composant reposant sur l'apprentissage machine est présenté en Figure 4.1.

Les questions principales portant sur une telle configuration sont les suivantes :

(1) Est-ce que le système oublie ce qu'il a initialement appris ou les connaissances auxquelles il avait accès initialement lorsqu'il s'adapte au cours du temps - concept connu sous le nom d'oubli catastrophique dans le cadre de l'apprentissage continu [Parisi et al., 2019] ?

(2) Est-ce que le système est en effet capable d'apprendre/d'accumuler les nouveaux éléments qui apparaissent en production, est-ce qu'il s'améliore lorsqu'il s'adapte ?

(3) Est-ce que le système arrive à inférer des connaissances supplémentaires à partir des nouveaux éléments récupérés en production dans le but de généraliser ?

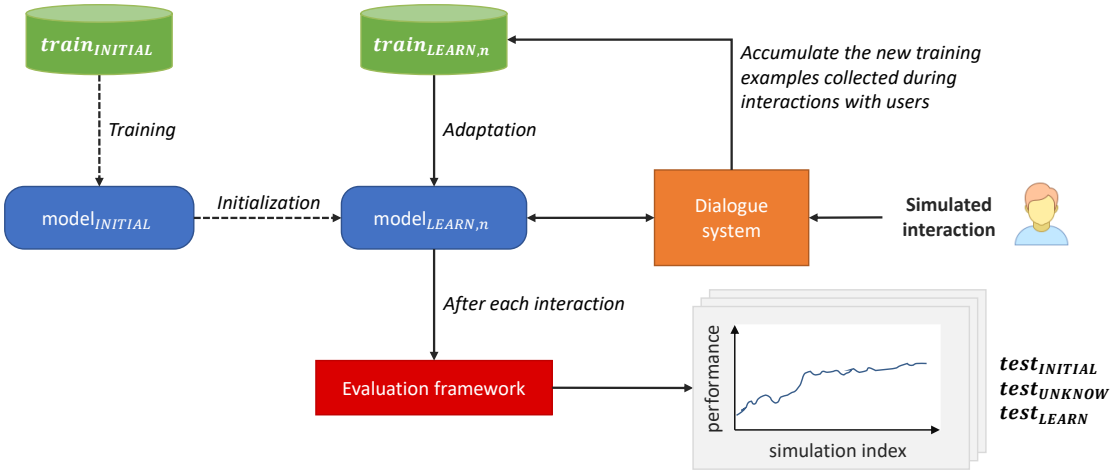


FIGURE 4.1 – Schéma général de la méthodologie d'évaluation pour l'apprentissage sur le terrain. Au cours des interactions du système de dialogue avec un utilisateur simulé, le système fait appel à un modèle $model_{LEARN,n}$ qui est adapté au cours du temps grâce aux nouveaux exemples d'entraînement collectés par le système. $model_{LEARN,n}$ est ensuite évalué sur les trois jeux de données de test après chaque interaction.

Afin de permettre une comparaison des différents états du système au cours du temps, les performances du système sur la tâche considérée doivent être mesurées en utilisant toujours le même jeu de données d'évaluation. Nous proposons d'utiliser en réalité trois jeux de données d'évaluation différents afin de répondre respectivement aux trois questions précédentes :

— $test_{INITIAL}$ composé de données qui sont similaires aux données initiales, afin d'évaluer l'oubli.
— $test_{LEARN}$ composé de données qui contiennent les éléments que le système doit apprendre/accumuler en production afin d'évaluer l'apprentissage en lui-même.
— $test_{UNKNOWN}$ composé d'éléments qui ne sont pas présents dans les données initiales et qui n'apparaîtront que pendant la phase de production afin d'évaluer les capacités de généralisation du système.

Concernant le système, son adaptation au cours du temps devra se faire de manière autonome, ce qui signifie que le système doit s'adapter sans l'aide d'expert du domaine ou de développeurs. De plus les interactions avec les utilisateurs et la collecte de leurs retours et corrections doivent se faire de la manière la plus naturelle possible. Ceci est nécessaire si l'on veut que l'utilisateur continue d'aider le système à s'améliorer et que le processus d'adaptation soit réussi. Ainsi, les seules entrées acceptées durant l'évaluation du système correspondent à des interactions avec les

utilisateurs en langage naturel. De plus, dans le cadre d'un scénario réel, il est impossible de définir à l'avance les jeux de données $test_{LEARN}$ et $test_{UNKNOWN}$. Pour cela, la méthodologie d'évaluation repose sur l'utilisation d'un utilisateur simulé - méthode commune pour l'évaluation des systèmes de dialogue [Deriu et al., 2020] - dans le but de simuler la phase de production. Ainsi, les données utilisées pour la simulation doivent être similaires à $test_{INITIAL}$ et $test_{LEARN}$ mais doivent être complètement séparées de $test_{UNKNOWN}$.

L'aspect continu de l'apprentissage sur le terrain doit aussi être évalué. En effet, il est problématique si l'utilisateur doit par exemple toujours corriger le système pour la même erreur et attendre que le système s'adapte. En conséquence, les performances du système sont ici évaluées après chaque interaction avec un utilisateur.

Pour résumer, la méthodologie d'évaluation présentée ici consiste en la simulation d'un utilisateur et l'évaluation après chaque interaction des performances du système courant sur $test_{INITIAL}$, $test_{LEARN}$ et $test_{UNKNOWN}$ pour respectivement évaluer l'oubli, l'apprentissage des éléments fournis en production et les capacités de généralisation. Nous considérons que cette méthodologie est adaptée à n'importe quel système de dialogue apprenant sur le terrain, à chacun de ses composants et à n'importe quel domaine.

4.4 Un système de dialogue apprenant sur le terrain

Afin d'éprouver la méthodologie d'évaluation présentée précédemment, nous avons implémenté un système de dialogue orienté tâche dans le domaine de la cuisine. Celui-ci est capable d'améliorer en continu et de manière autonome son module de compréhension (NLU) lors de ses interactions avec ses utilisateurs (apprentissage sur le terrain). Le but de ce système est de trouver et de proposer des recettes de cuisine qui correspondent aux critères indiqués par l'utilisateur. Le système repose notamment sur une base de connaissances (KB) contenant des informations sur des recettes et sur des aliments. Le fonctionnement général du système est décrit en Figure 4.2. À noter que l'objectif ici n'est pas de proposer un système performant, mais simplement un système présentant la majorité des caractéristiques attendues d'un système apprenant sur le terrain. Ce travail permet de se rendre compte des défis apportés par l'apprentissage sur le terrain du point de vue de l'élaboration d'un tel système et de son évaluation.

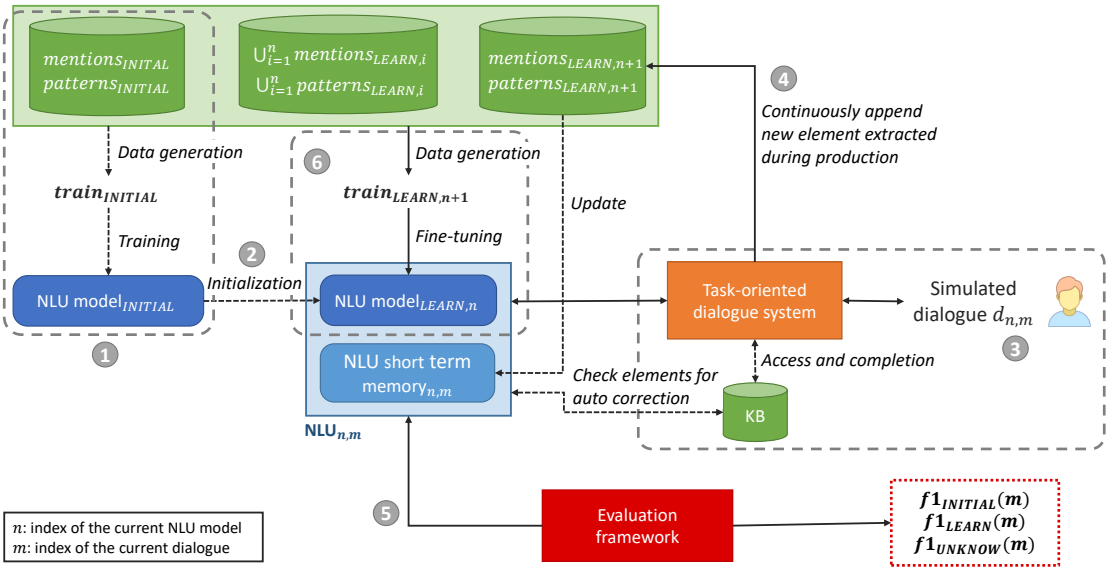


FIGURE 4.2 – Fonctionnement général du système et évaluation pendant la phase de production (simulée). 1) Entraînement du modèle initial sur $train_{INITIAL}$, 2) Initialisation du modèle de compréhension de la langue, 3) Interactions avec l'utilisateur simulé, 4) Extraction continue de nouvelles connaissances (mentions, patrons, exemple d'entraînement) et mise à jour de la mémoire à court terme, 5) Évaluation sur $test_{INITIAL}$, $test_{LEARN}$, $test_{UNKNOWN}$ et $test_{REAL}$ à la fin de chaque dialogue et 6) Si une des conditions pour l'adaptation du modèle est validée, génération de $train_{LEARN,n}$ à partir des anciens et des nouveaux mentions et patrons et fine-tuning du modèle.

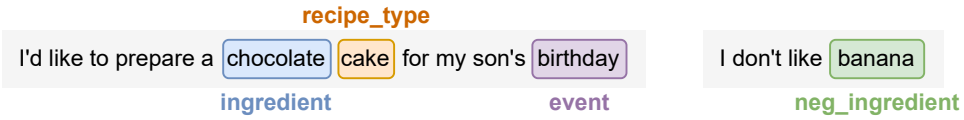


FIGURE 4.3 – Exemples d'énoncés utilisateur annotés avec des slots adaptés au domaine de la cuisine.

4.4.1 Description de la tâche et du modèle

Tâche. Nous nous concentrons ici sur l'amélioration du composant de compréhension de la langue (NLU). Comme présenté en section 2.2.2, le composant de NLU se divise généralement en deux tâches : la détection des slots et la détection de l'intention. Nous ne considérons ici que l'amélioration de la tâche de détection des slots afin de simplifier les expériences. De plus, nous considérons cette tâche comme plus intéressante à étudier puisque elle est plus complexe que la tâche de détection de l'intention. La tâche de détection des slots a pour but d'identifier les slots apparaissant dans un énoncé utilisateur au cours du dialogue, un slot correspondant à une paire (type du slot, valeur du slot). Dans ce chapitre, nous parlerons aussi de « mention » pour désigner la valeur d'un slot. Cette tâche peut être considérée comme une tâche d'annotation de séquence où chaque mot de l'énoncé utilisateur doit être associé à une étiquette au format IOB (voir section 2.2.2). Dans le cadre de ces expériences, nous avons décidé de travailler sur le domaine de la cuisine et avons défini 7 types de slots : *recipe type, ingredient, preparation technique, origin, meal, event* et *other category*. Par exemple, étant donné l'énoncé utilisateur suivant : `"I'd like to prepare a chocolate cake for my son's birthday"`, les slots à détecter sont `(ingredient, 'chocolate')`, `(recipe type, 'cake')` et `(event, 'birthday')`. Pour prendre en compte les contextes négatifs, nous avons ajouté les variantes négatives de certains types de slots. Des exemples sont illustrés en Figure 4.3.

Modèle. La tâche de détection des slots est aujourd'hui majoritairement réalisée à l'aide de modèles d'apprentissage profond. Dans le cadre de ces expériences, nous utilisons une architecture basée sur un LSTM bidirectionnel (BiLSTM). En effet, ce type de réseaux de neurones est adapté à l'annotation de séquence puisqu'il permet de prendre en compte pour chaque token la sortie de chaque couche correspondant aux tokens situés à sa droite et à sa gauche [Dyer et al., 2015]. Plus précisément, notre modèle consiste en une couche de plongements lexicaux (*word embeddings*), suivie d'un BiLSTM de deux couches de tailles 128, suivies d'une couche linéaire. La couche de plongements lexicaux est initialisée par des vecteurs de tailles 300 entraînés à l'aide de la méthode Word2Vec CBOW [Mikolov et al., 2013] sur les données Wikipédia en anglais [Ghannay et al., 2020]. L'entraînement est optimisé à l'aide de l'optimiseur Adam avec un taux d'apprentissage initial de 10^{-3} et via la minimisation de l'entropie croisée des prédictions du modèle. Avant de simuler la phase de production, nous entraînons le modèle sur $train_{INITIAL,TRN}$. La validation est faite en sélectionnant le modèle dont le F1-score est le plus haut sur le jeu de données de développement $train_{INITIAL,DEV}$.

4.4.2 Collecter des nouvelles connaissances

Détecter la présence de nouvelles connaissances

Pour construire le système de dialogue apprenant sur le terrain, l'idée principale est de tirer profit de ses interactions avec ses utilisateurs afin de collecter de nouveaux exemples d'entraînement pour adapter au cours du temps le modèle de détection des slots. La première étape consiste à détecter qu'une ou des nouvelles connaissances peuvent être apprises. Dans notre cas, nous faisons l'hypothèse que des nouvelles connaissances peuvent toujours être extraites de l'énoncé courant de l'utilisateur si l'énoncé contient des slots. L'hypothèse est vérifiée par le système après l'extraction et l'identification des connaissances.

Annoter l'énoncé utilisateur initial

La prochaine étape consiste en l'extraction et l'identification des nouvelles connaissances, qui consiste ici en l'annotation de l'énoncé utilisateur initial. Cette étape repose sur la détection *a posteriori* qu'une erreur de compréhension a été commise, de manière similaire à [Hancock et al., 2019]. En réponse au premier énoncé utilisateur, le système transmet en langage naturel les sorties de son modèle correspondant aux slots qu'il a détectés. Si l'utilisateur continue la conversation telle qu'attendue par le système, celui-ci suppose que la sortie du modèle est correcte (principe de progressivité [Albert and Ruiter, 2018]) ; au contraire, si le système détecte que l'utilisateur le notifie que sa requête a été mal comprise, la sortie du modèle est considérée comme étant incorrecte. Si la sortie du modèle est considérée comme correcte, l'énoncé utilisateur est annoté avec les slots détectés et l'énoncé annoté est stocké dans une mémoire temporaire. À l'inverse, si la sortie est considérée comme incorrecte, le système essaie de corriger ses prédictions et répond à l'utilisateur conformément à ses corrections, toujours en transmettant les slots qu'il a détectés en langage naturel. Comme précédemment, si l'utilisateur continue la discussion, il indique de manière indirecte que la correction est correcte. Sinon, cela signifie que le système n'est pas capable d'extraire de nouvelles connaissances à partir de l'énoncé utilisateur initial et le dialogue s'arrête.

Pour corriger la sortie initiale du modèle, nous faisons ici l'hypothèse que l'utilisateur va paraphraser sa requête initiale lorsqu'il indique au système qu'une erreur a été commise (par exemple `"You misunderstood me, I asked for a cake recipe without eggs"`). Ce comportement est en effet habituel dans le cadre d'échange humain/humain bien qu'il faille noter que ce n'est pas le seul comportement possible [Hough, 2014] et que le comportement d'un humain face à un système de dialogue peut différer de celui entre deux humains. À partir de cette hypothèse, le système compare la requête initiale avec la paraphrase en re-

tirant les mots vides (*stop words* an anglais) et en ne gardant que les segments de tokens en commun. Par exemple, si l'on considère la requête initiale `"Do you have cake recipes for people allergic to eggs?"` et la paraphrase précédente `"You misunderstood me, I asked for a cake recipe without eggs"`, les segments de tokens en commun restants sont `cake` et `eggs`. Ensuite, le système vérifie dans sa KB si un élément associé à chaque segment de tokens récupéré existe et essaie d'extraire le type de slot associé à cet élément grâce aux méta-données disponibles sur cet élément. Si un type de slot est extrait, l'énoncé initial de l'utilisateur peut être annoté. Cette étape est cruciale mais dépend fortement de la qualité de la KB et de sa capacité à faire des liens entre les informations qu'elle contient et les types de slots à détecter.

Nouvelles mentions et nouveaux patrons

Puisque l'apprentissage sur le terrain implique l'hypothèse d'un monde ouvert, des nouvelles entités qui ne sont pas originellement dans la KB peuvent en pratique apparaître au cours du temps dans les énoncés utilisateur. Nous pouvons imaginer, par exemple, que la KB associée à notre système de dialogue a été construite uniquement à partir de données de cuisine des pays occidentaux, de telle manière qu'un ingrédient tel que le liseron d'eau, qui est originaire d'Extrême Orient, ne soit pas originellement contenu dans la KB. Ainsi, les performances du modèle de détection des slots peuvent être impactées négativement puisque le modèle devra détecter des nouvelles mentions jamais vues pendant l'entraînement initial. Pour cette raison, nous avons décidé d'incorporer ce phénomène dans nos expériences. Cependant, étant donné que le système se repose sur les éléments de sa KB pour corriger les prédictions de son modèle de compréhension, il est nécessaire de permettre à la KB d'évoluer au cours du temps afin de prendre en compte ces nouvelles entités. Pour ce faire, si aucun élément de la KB ne correspond aux critères de recherche détectés par le modèle NLU ou issus de la correction de la prédiction, alors le système a la possibilité d'extraire via une source d'informations non structurée des nouvelles recettes contenant ces critères de recherche. Ceci permet la complétion au cours du temps de la KB. Dans le cadre de nos expérience, l'extraction des nouveaux éléments de la KB est simulée afin de nous permettre de ne pas trop complexifier le système. Il est cependant possible d'imaginer que cette extraction se fasse automatiquement au moment où le dialogue a lieu, via des documents textuels non structurés extraits du web, de manière similaire à [Komeili et al., 2022].

L'ensemble des processus décrits ici nous permet ainsi d'annoter l'énoncé initial de l'utilisateur et possiblement de compléter la KB avec des éléments supplémentaires, se traduisant par l'introduction de nouvelles mentions. De plus, à partir d'un

énoncé annoté, il est possible d'inférer un patron de phrase. Par exemple, à partir de l'énoncé `"I've been invited to a brunch. Any suggestion?"`, peut être déduit le patron de phrase `"I've been invited to a $event. Any suggestion?"`. Ainsi, plusieurs nouveaux exemples d'entraînement peuvent être générés à partir de ce patron de phrase et des mentions associées aux types de slots contenus dans le patron qui peuvent être récupérées dans la KB. De la même manière, l'ajout d'une nouvelle mention dans la KB permet la génération de nouveaux exemples d'entraînement.

Pour résumer, les nouvelles connaissances pouvant être collectées au cours des interactions avec les utilisateurs consistent en de nouvelles mentions, de nouveaux patrons et de nouveaux exemples d'entraînement (si le patron est déjà connu).

4.4.3 Adapter la compréhension de la langue

Pour adapter le modèle NLU au cours du temps, le modèle courant est adapté par fine-tuning sur les nouveaux exemples d'entraînement générés à partir des connaissances collectées précédemment ($train_{LEARN,n}$ sur la Figure 4.2). Plus précisément, un nouveau modèle $model_{LEARN,n+1}$ est initialisé avec les poids du modèle courant $model_{LEARN,n}$ et est entraîné sur $train_{LEARN,n+1}$. Les hyperparamètres restent les mêmes qu'au cours de l'entraînement initial. L'adaptation est déclenchée si le nombre de nouvelles mentions, le nombre de nouveaux patrons ou le nombre de nouveaux exemples d'entraînement dépasse une valeurs propre à chaque type de d'élément. Pour éviter le phénomène d'oubli [Parisi et al., 2019], nous utilisons une méthode simple appelé *replay* [Robins, 1995] afin de rejouer des anciens exemples d'entraînement lors du fine-tuning du modèle sur de nouveaux exemples. En pratique, toutes les données d'entraînement sont générées à partir de listes de patrons de phrases et de listes de mentions. Ainsi, $train_{LEARN,n+1}$ est généré à partir des patrons et des mentions de l'entraînement initial, des patrons et mentions collectés pour des adaptations passées et des patrons et mentions nouvellement collectées. Pour chaque adaptation le système génère 1000 exemples d'entraînement et ajoute les nouveaux exemples d'entraînements uniques collectés au cours des interactions. Deux types de replay sont testés ici : 1) le Replay avec Patrons et Mentions (RPM) [1] et 2) le Replay avec uniquement les Mentions (RM) [2].

Cependant, puisque l'aspect continu de l'apprentissage sur le terrain est important

1. Le système génère 25% de nouveaux exemples avec des nouveaux patrons et nouvelles mentions, 25% avec des nouveaux patrons et anciennes mentions, 25% avec des anciens patrons et nouvelles mentions et 25% avec des anciens patrons et anciennes mentions.

2. Le système génère les exemples d'entraînement avec les nouveaux patrons et les nouvelles mentions, sauf pour les types de slots pour lesquels le système n'a pas pu collecter de nouvelles mentions.

et puisqu'un modèle d'apprentissage tel que celui que nous utilisons nécessite du temps pour être adapté, une mémoire à court terme doit être ajoutée. Ainsi, le composant de compréhension peut être adapté de deux manières : une adaptation continue avec la mémoire à court terme (*short term memory* sur la Figure 4.2), et une adaptation incrémentale avec l'adaptation du modèle NLU représentant ici la mémoire à long terme. Dans le cadre de ces expériences, seules les nouvelles mentions sont considérées par la mémoire à court terme. En effet, nous considérons qu'un utilisateur attend du système de se rappeler tout de suite d'une nouvelle mention introduite dans un dialogue précédent et que ceci est caractéristique d'un apprentissage sur le terrain. Pour un patron à l'inverse, un utilisateur peut être moins exigeant. À chaque fois que le modèle NLU est adapté, la mémoire à court terme est ré-initialisée. La présence de deux mémoires nécessite que le modèle combine et arbitre les slots détectés par le modèle NLU et la mémoire à court terme. La stratégie suivie est détaillée dans l'algorithme 1. Celle-ci a été définie de manière empirique au cours d'expériences préliminaires. Nous avons observé en particulier que la prise en compte du contexte négatif d'une mention impacte considérablement les performances.

Algorithm 1 Mise à jour des sorties du modèle de détection des slots (NLU) à l'aide de la mémoire à court terme (*short term memory*, STM)

Input : user's utterance u
Output : slots detected in u

procedure GET_SLOTS(u)
  $slots_{model} = get\_slots\_from\_model(u)$
  $slots_{stm} = get\_slots\_from\_stm(u)$
  initialize *slots* as an empty list of slots
  for s_{model} in $slots_{model}$ **do**
    for s_{stm} in $slots_{stm}$ **do**
      if $equal(s_{model}, s_{stm})$ **then**
        add s_{model} to *slots*
      else if $equal(s_{model}.value, s_{stm}.value)$ and $is\_negative(s_{model}.type)$ **then**
        add ($negative(s_{stm}.type)$, $s_{stm}.value$) to *slots*
      else if $s_{model}.value$ in $s_{stm}.value$ **then**
        add s_{stm} to *slots*
      else if $s_{stm}.value$ in $s_{model}.value$ **then**
        if $s_{model}.value$ in KB **then**
          add s_{model} to *slots*
        else
          add s_{stm} to *slots*
    if nothing added to slots **then**
      add s_{model} to *slots*
  return *slots*

4.5 Contexte expérimental

Pour évaluer l'apprentissage sur le terrain, nous simulons la phase de production à l'aide d'un utilisateur simulé et de la simulation de la complétion de la KB. L'utilisateur est simulé notamment à partir d'un jeu de données annotées nommé *simulation*. L'utilisateur simulé, la préparation des données et les conditions d'évaluation sont présentés dans cette section.

4.5.1 Simulation de l'utilisateur

Comme présenté en section 4.3, un utilisateur simulé est utilisé pour simuler la phase de production afin d'évaluer le système. L'utilisateur simulé consiste ici en un système à base de règles. Seul le premier énoncé de l'utilisateur est généré, celui-ci provenant du jeu de données annotées *simulation*. L'utilisateur simulé suit le scénario suivant :

1. L'utilisateur demande au système de lui fournir une recette étant donnés certains critères - par exemple le type de recette et/ou un ingrédient.
2. Le système lui donne une réponse en précisant les critères qu'il a identifiés, c'est-à-dire les slots prédits.
3. Si les slots prédits ne sont pas ceux qui correspondent aux annotations de référence, l'utilisateur notifie le système de son erreur via le patron de phrase `"You misunderstood me. I want a recipe with <slots>"` avec les slots énoncés en langue naturelle à la place de `<slots>` (par exemple `"I want a recipe with brunch as event"`). Sinon, l'utilisateur se comporte normalement.
4. Si le système détecte qu'il s'est trompé, il corrige sa prédiction des slots et répond à l'utilisateur en accord avec sa correction tout en précisant les slots identifiés après correction.
5. Si les slots corrigés correspondent à la référence, alors l'utilisateur se comporte normalement, sinon il termine la conversation.

Il faut noter ici qu'avec cet utilisateur simulé simplifié, détecter qu'une erreur a été commise par le système pour enclencher la correction devient trivial.

Les interactions entre l'utilisateur simulé et le système ainsi que les actions principales du système sont résumées dans la Figure 4.4.

4.5.2 Préparation des données

Plusieurs jeux de données annotées sont nécessaires pour entraîner et évaluer le système de dialogue apprenant sur le terrain. Ces jeux de données sont les suivants :

$train_{INITIAL,TRN}$ (20k), $train_{INITIAL,DEV}$ (4k), *simulation* (20k), $test_{INITIAL}$ (1k), $test_{LEARN}$ (1k) et $test_{UNKNOWN}$ (1k). Comme il n'existe aucun corpus de données annotées pour la tâche de détection des slots dans le domaine de la cuisine, les données annotées ont été générées à l'aide de patrons de phrases et de listes de mentions comme illustré en Figure 4.5. Les patrons de phrases ont été définis manuellement dans le cadre du projet de recherche LIHLITH (voir section 1.2) et les mentions ont été extraites à partir d'internet. La génération des données à partir de patrons et de mentions nous permet aussi d'avoir plus de contrôle sur les nouveaux éléments à apprendre au cours de la simulation de la phase de production. Pour ce faire, l'ensemble des patrons et l'ensemble des mentions sont divisés en plusieurs sous-ensembles nommés $INITIAL$, $LEARN$ et $UNKNOWN$ pour ensuite générer les différents jeux de données. Ainsi, 25% des patrons sont mis dans $INITIAL$, 60% dans $LEARN$ et les 15% restants dans $UNKNOWN$, tandis que 60% des mentions sont mises dans $INITIAL$, 25% dans $LEARN$ et les 15% restantes dans $UNKNOWN$. La proportion de patrons dans $LEARN$ est plus importante que pour les mentions, parce qu'on estime qu'un utilisateur peut exprimer une même intention de beaucoup de manières différentes tandis qu'il aura moins de variations au niveau des mentions utilisées.

Pour *simulation*, pour refléter au plus près la réalité, nous faisons l'hypothèse que les probabilités qu'un nouveau patron ou qu'une nouvelle mention apparaissent dans un énoncé utilisateur sont respectivement égales à 0,7 et 0,3. La différence entre les deux probabilités s'explique de la même manière que précédemment. Ainsi, le jeu de données *simulation* est construit à partir de $INITIAL$ et $LEARN$ avec des distributions de patrons et de mentions qui suivent l'hypothèse précédente.

Pour le jeu de données de développement $train_{INITIAL,DEV}$ utilisé pendant l'entraînement initial du modèle NLU pour sélectionner le meilleur modèle, nous avons fait la même supposition et divisé l'ensemble $INITIAL$ en deux ensembles disjoints, tels que $train_{INITIAL,DEV}$ est composé de mentions et de patrons présents dans $train_{INITIAL,TRN}$ et d'autres non, afin de sélectionner le modèle capable au mieux de généraliser dans un monde ouvert. Puisque tous les jeux de données de test pour l'évaluation sont générés automatiquement, nous avons ajouté un jeu de données de test réel $test_{REAL}$ (744 requêtes annotées), composé de questions relatives au domaine de la cuisine [3].

Dans le cadre des expériences, la complétion de la KB est simulée. Pour ce faire, lors de la préparation des données, les mentions de $LEARN$ et les recettes associées à ces mentions sont retirées de la KB. Pendant ses interactions avec l'utilisateur, lorsque le système décide de rechercher des informations sur un élément, si celui-ci est en effet contenu dans $LEARN$, alors l'élément ainsi que les recettes associées

3. Ce jeu de données a été annoté et collecté dans le cadre du projet LIHLITH.

seront ajoutées à la KB. Dans le cadre de ces expériences, nous avons retiré 40% des ingrédients en partant des ingrédients les moins fréquents dans les recettes de la KB, si bien que les mentions de $LEARN$ consistent uniquement en des mentions correspondant à des noms d'ingrédients.

4.5.3 Évaluation

Pour évaluer les performances du composant de compréhension, le F1-score est calculé sur les jeux de données de test à l'aide du script `conlleval.pl` [Tjong Kim Sang and Buchholz, 2000] à la fin de chaque dialogue simulé et après l'adaptation du composant de compréhension. Afin de permettre la reproduction des expériences, nous fixons la valeur de la graine aléatoire (*random seed*) lors de la préparation des données. À chaque fois, un modèle NLU initial est entraîné et deux simulations sont conduites avec les deux types de replay présentés précédemment.

4.6 Résultats et discussion

La Figure 4.6 montre l'évolution des performances du composant de compréhension au cours de la simulation de la phase de production sur les différents jeux de test avec les deux méthodes d'adaptation du modèle NLU (RPM et RM) [4].

Évolution des performances sur les différents jeux de données de test

Les performances sur $test_{INITIAL}$ restent stables au cours du temps pour atteindre à la fin un score inférieur à celui du modèle initial de seulement 0,5 points (voir tableau 4.1), ce qui montre que les méthodes utilisées permettent d'éviter l'effet d'oubli.

Sur $test_{LEARN}$, les deux performances augmentent de manière significative au début, ce qui est dû à la manière dont le jeu de données *simulation* a été construit : comme la génération se fait de manière aléatoire et que les énoncés ne sont pas triés, alors la majorité des nouveaux éléments sera déjà apparue au moins une fois au bout de quelques dialogues simulés, ce qui semble arriver dans notre cas au bout du 4 000^{e} dialogue. De plus, les deux méthodes semblent ensuite atteindre un plateau, ce qui montre aussi les limitations d'un protocole expérimental se reposant sur des données générées. Cependant, nous pouvons observer que RM a de meilleurs résultats par rapport à RPM, pouvant aller jusqu'à une différence de 2 points et que cette différence diminue progressivement de telle manière que les

4. Nous avons réalisé les mêmes expériences avec des données générées avec d'autres valeurs de graines aléatoires. Comme nous avons observé des résultats similaires, nous ne présentons ici que les résultats obtenus avec une graine aléatoire.

performances avec les deux méthodes sont similaires à la fin. Cette observation suggère que l'apprentissage de nouvelles mentions est plus profitable au modèle lors de l'adaptation de celui-ci puisque dans le cas de la méthode RM, les nouvelles mentions sont en moyenne deux fois plus présentes dans $train_{LEARN,n}$ que dans le cas de la méthode RPM. En effet, pour la méthode RPM la moitié des données générées comprennent au moins une nouvelle mention, tandis que pour la méthode RM toutes les données générées comprennent au moins une nouvelle mention.

Sur $test_{UNKNOWN}$, l'adaptation via la méthode RM semble donner des performances plus stables qu'avec la méthode RPM et la différence de scores peut atteindre les -3,5 comme les +3,5 points. Cependant, cette différence n'est pas visible dans le tableau 4.1 lorsque l'on observe les résultats finaux sur $test_{UNKNWON}$.

De manière globale, nous constatons que l'observation de l'évolution des performances au cours du temps permet de se rendre compte de phénomènes et de différences entre les deux méthodes, qui ne sont pas observables seulement à partir des performances finales. Ceci montre l'intérêt de l'évaluation au cours de l'apprentissage sur le terrain de la méthodologie d'évaluation définie.

Impact de la mémoire à cours terme

Comme le nombre d'adaptations du modèle NLU dans le cadre de ces expériences est assez élevé (322/327 pour RPM/RM), l'évolution continue n'est pas visible sur la Figure 4.6. Ainsi, nous avons calculé la différence de score entre adaptation, c'est-à-dire entre chaque modèle $model_{LEARN,i}$ et $model_{LEARN,i+1}$, afin d'analyser l'impact de la mémoire à court terme dans nos expériences. Nous observons que la mémoire à court terme améliore au maximum les performances de 0,43 points (entre les dialogue n°318 et n°390 avec 16 nouvelles mentions) et dégrade les performances de 0,04 points au maximum sur $test_{LEARN}$ avec la méthode RPM (moyenne : 0,01, médiane : 0).

Comparaison des performances à la fin de la simulation

Par la suite, nous avons comparé les performances finales de modèles issus de l'apprentissage sur le terrain ($model_{LEARN}$) avec celles d'autres modèles, comme présenté dans le tableau 4.1.

Si nous comparons les performances finales sur $test_{LEARN}$ des modèles issus de l'apprentissage sur le terrain $model_{LEARN,RPM,n=N}$ et $model_{LEARN,RM,n=N}$ avec celles de modèle initial $model_{INITIAL}$, nous pouvons constater que le composant de compréhension est amélioré sur le terrain via ses interactions avec l'utilisateur simulé et grâce aux méthodes de collecte et d'adaptation décrites en section 4.4.

model	$test_{INITIAL}$	$test_{LEARN}$	$test_{UNKNOWN}$	$test_{WEIGHTED}$	$test_{REAL}$
$model_{INITIAL}$	98.99	89.33	68.26	82.83	36.18
$model_{LEARN,STM}$	98.50 (-0.49)	91.61 (+2.28)	67.52 (-0.74)	83.36 (+0.53)	-
$model_{LEARN,RPM,n=N}$	98.56 (-0.43)	97.48 (+8.15)	71.11 (+2.85)	87.15 (+4.32)	37.16 (+0.98)
$model_{LEARN,RM,n=N}$	99.14 (-0.15)	97.63 (+8.30)	70.79 (+2.53)	87.20 (+4.37)	37.70 (+1.52)
$model_{SIMU}$	99.94 (+0.95)	99.60 (+10.27)	71.49 (+3.23)	88.42 (+5.59)	41.24 (+5.06)

TABLEAU 4.1 – F1-scores de différents modèles sur différents jeux de données de test. L'indice N correspond à l'index de la dernière adaptation du modèle à la fin de la simulation dans le cas où le modèle a été adapté sur le terrain (ici $N = 322$ et $N = 327$ respectivement pour les méthodes RPM et RM.). $model_{LEARN,STM}$ correspond à $model_{INITIAL}$ adapté uniquement avec la mémoire à court terme (pas de fine-tuning du modèle pendant l'évaluation). $model_{SIMU}$ correspond à $model_{INITIAL}$ adapté par fine-tuning sur le jeu de données *simulation*. Les entraînements et les évaluation ont été faits avec des données générées avec une valeur de graine aléatoire de 1.

Afin d'observer l'intérêt de la mémoire à court terme, nous avons lancé une simulation durant laquelle l'adaptation se fait seulement avec la mémoire à court terme, tel que le modèle NLU initial n'est jamais adapté au cours du temps. Les performances finales du modèle NLU initial, augmenté de la mémoire à court terme mise à jour durant la simulation, sont données dans le tableau 4.1 et correspondent au modèle $model_{LEARN,STM}$. La comparaison des performances du modèle $model_{LEARN,STM}$ avec celles de $model_{INITIAL}$ montre que la mémoire à court terme est bénéfique au système. Cependant, les comparaisons avec $model_{LEARN,RPM,n=N}$ et $model_{LEARN,RM,n=N}$ montrent que la mémoire à court terme seule n'est pas suffisante.

Par la suite, afin d'obtenir un aperçu des performances optimales d'un modèle entraîné sur les données de *simulation*, nous avons adapté $model_{INITIAL}$ par fine-tuning directement sur les données annotées de *simulation*, sans simulation de la phase de production. Le modèle résultant est nommé $model_{SIMU}$. Nous comparons ensuite les performances de $model_{SIMU}$ avec les performances finales de $model_{LEARN,RPM,n=N}$ et $model_{LEARN,RM,n=N}$. Nous observons que les scores sont proches sur $test_{INITIAL}$ et $test_{UNKNOWN}$, mais que la différence est de presque 2 points sur $test_{LEARN}$ et d'environ 3,5 points sur $test_{REAL}$ en faveur de $model_{SIMU}$. Ceci montre que certains éléments contenus dans les données *simulation* n'ont pas pu être extraits par le système au cours de la simulation lors de l'apprentissage sur le terrain.

À partir des journaux du système, nous observons notamment que seuls 0,14% des énoncés utilisateurs initiaux ont pu être annotés correctement grâce la correction

du système. Pour rappel, la correction a lieu lorsque l'utilisateur a paraphrasé sa requête initiale suite à une mauvaise prédiction des slots de l'énoncé par le système. Il faut noter ici que les nouvelles mentions à apprendre pendant l'entraînement concernent uniquement les noms d'ingrédients. Cependant, le modèle peut quand même être amené à faire des erreurs sur d'autres types de slots, tels que les types de recettes ou les événements. De fait, nous pouvons supposer que l'écart de performance sur $test_{LEARN}$ est dû en partie au fait que l'état de la KB ne permet pas de corriger correctement les énoncés utilisateur, aussi bien à cause de l'extraction du type de slot qui n'est pas toujours possible, que parce que certaines mentions ne peuvent pas être associées à une entité dans la KB. De plus, comme l'entraînement du modèle initial se fait sur des données générées à partir d'un sous-ensemble des patrons, le modèle initial aura rencontré les différents types de slots dans moins de contextes différents que $model_{SIMU}$, ce qui peut impacter négativement tous les types de slots et pas seulement les ingédients.

Enfin, nous observons que les performances des différents modèles sur le jeu de données réelles $test_{REAL}$ sont bien inférieures aux performances sur les données générées, ce qui montre de nouveau la limitation d'une évaluation se reposant sur des données générées.

4.7 Conclusion

Dans ce chapitre, nous avons décrit une première version d'une méthodologie générale pour l'évaluation continue de l'apprentissage sur le terrain d'un système de dialogue. À notre connaissance, celle-ci correspond à la première méthodologie d'évaluation de ce type. Cette méthodologie consiste de manière globale à simuler les interactions avec les utilisateurs et d'évaluer après chaque interaction les performances du système courant sur les jeux de données suivants : $test_{INITIAL}$, données similaires aux données initiales pour évaluer l'oubli ; $test_{LEARN}$, données contenant les éléments devant être appris au cours de la simulation ; et, $test_{UNKNOWN}$, données contenant des éléments n'apparaissant ni dans les données initiales, ni dans les données de simulation pour évaluer la généralisation. Nous avons aussi construit un système de dialogue orienté tâche qui peut améliorer de manière autonome et continue son module de compréhension grâce à ses interactions avec ses utilisateurs auquel nous avons appliqué la méthodologie d'évaluation définie plus tôt. Les expériences nous ont permis de voir les limitations d'une évaluation reposant sur des jeux de données de test générés à partir de patrons de phrases et de liste de mentions. Cependant, cette méthodologie nous a permis de comparer au cours du temps les différentes méthodes d'adaptation et d'identifier certaines de leurs différences, tandis que les protocoles d'évaluation déjà existants ne l'auraient pas

permis.

Dans le cadre de futures expériences, il serait intéressant d'adapter le système à l'apprentissage sur le terrain de nouveaux slots et de nouveaux domaines afin de se rapprocher d'un contexte expérimental de LL. Ceci peut se faire en combinant la méthodologie d'évaluation décrite ici avec les méthodologies d'évaluation décrites dans le cadre de l'apprentissage continu en ligne [Aljundi et al., 2019]. Il serait aussi intéressant d'ajouter à la méthodologie d'évaluation un moyen d'évaluer la robustesse de l'apprentissage sur le terrain du système face à du bruit, par exemple si l'utilisateur fournit une mauvaise correction au système.

Dans ce chapitre, nous avons décidé de simplifier la nature des nouveaux éléments pouvant être appris au cours du temps, afin de nous concentrer sur la conception d'un système complet auquel nous pourrions appliquer la méthodologie définie au préalable pour l'évaluation de l'apprentissage sur le terrain. Ainsi, dans la suite de ce mémoire, nous cherchons à complexifier la nature des nouveaux éléments en nous intéressant à l'introduction de nouvelles langues (chapitre 5), et à l'introduction de nouveaux types de slots via de nouveaux domaines (chapitre 6). Ces deux chapitres se concentrent de plus sur un autre aspect important du LL qui n'a pas pu être étudié dans ce chapitre et qui correspond à l'étude du transfert.

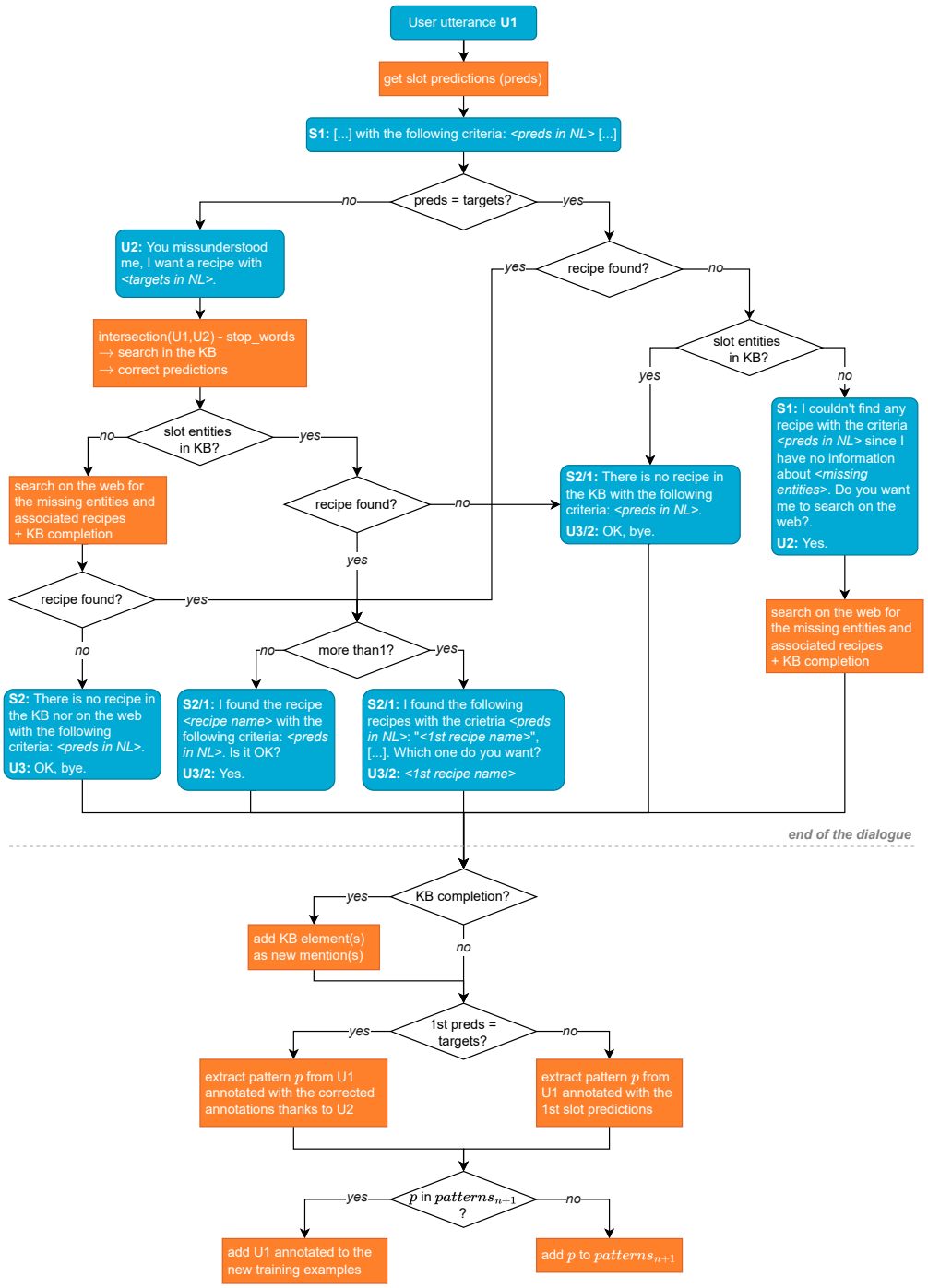


FIGURE 4.4 – Diagramme de décision représentant les interactions entre l'utilisateur simulé et le système accompagnées des actions principales du système. Les rectangles bleus aux bords arrondis illustrent les énoncés utilisateur Ui et système Si, avec i l'index du tour de dialogue. Les rectangles oranges illustrent les actions du système. Pour des raisons de lisibilité, le cas où les corrections des prédictions ne correspondent pas aux références n'est pas illustré ici. Dans ce cas, une mention peut être ajoutée si la KB a été complétée, mais aucun patron ne sera ajouté.

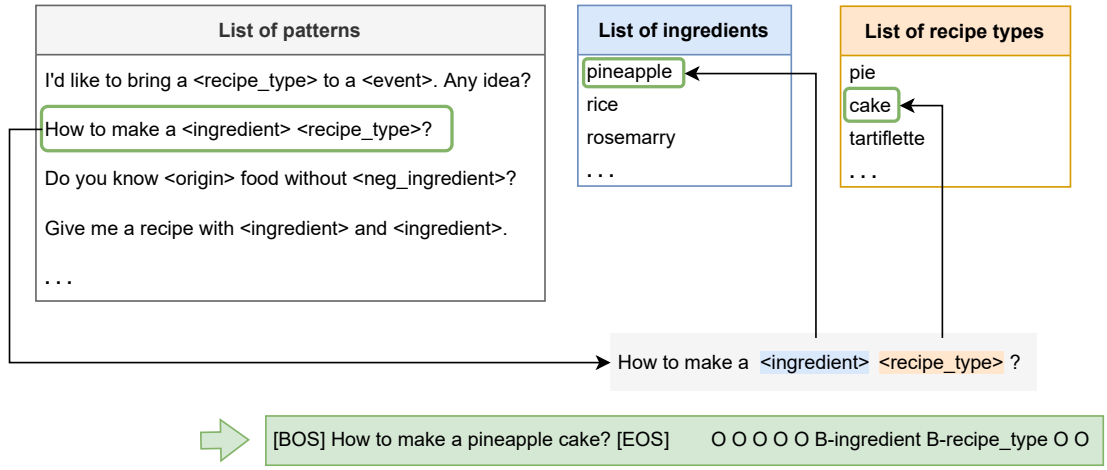


FIGURE 4.5 – Génération de données annotées pour la tâche de détection des slots à l'aide de patrons de phrases et de listes de mention selon les étapes suivantes : 1) Un patron de phrase est choisi aléatoirement dans la liste de patrons définis au préalable ; 2) Pour chaque type de slot apparaissant dans le patron, une mention est choisie aléatoirement dans la liste correspondante ; 3) Les types de slots apparaissant dans le patron sont remplacés par les mentions choisies et les annotations sont générées ; 4) L'opération est répétée jusqu'à ce que n énoncés annotés soient générés en faisant en sorte que tous les patrons apparaissent le même nombre de fois dans les données générées (à un près).

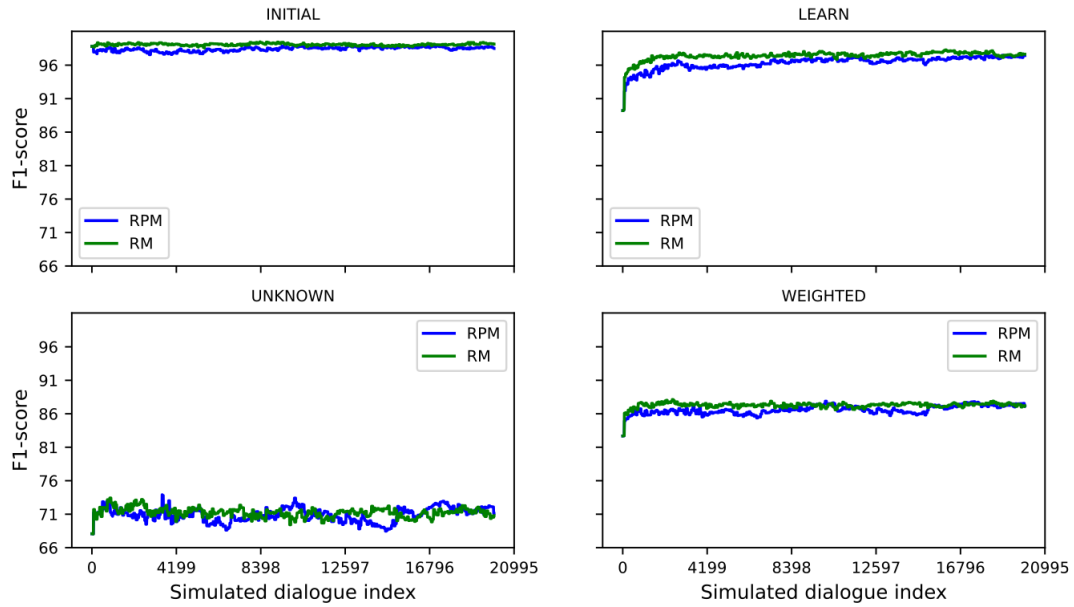


FIGURE 4.6 – Évolution du F1-score au cours de la simulation de la phase de production sur $test_{INITIAL}$, $test_{LEARN}$, $test_{UNKNOWN}$ et $test_{WEIGHTED}$ avec des données générées avec une valeur de graine aléatoire de 1 ($F1_{WEIGHTED} = F1_{INITIAL} * 0.2 + F1_{LEARN} * 0.4 + F1_{UNKNOWN} * 0.4$).

Chapitre 5

Étude du transfert entre langues dans le cadre de l'apprentissage continu

5.1 Introduction

Dans le chapitre précédent, nous nous sommes intéressés à l'apprentissage sur le terrain de nouvelles instances pour la tâche de détection des slots. De fait, nous n'avons pas pu étudier le transfert, bien que celui-ci soit un aspect important du LL. Ainsi, dans ce chapitre nous nous intéressons aux capacités de transfert des systèmes de dialogue. Comme présenté dans le chapitre 2, aujourd'hui, la plupart des modèles appliqués aux tâches associées aux systèmes de dialogue reposent sur des modèles issus de l'architecture `transformer`, pré-entraînés de manière auto-supervisée sur de larges corpus de données textuelles. Ces modèles pré-entraînés rendent possible le transfert de connaissances acquises lors de leur pré-entraînement vers une tâche cible, comme observé avec BERT [Devlin et al., 2019b], T5 [Raffel et al., 2020] ou GPT-3 [Brown et al., 2020b]. Des versions multilingues de ces modèles ont aussi été entraînées et ont, de la même manière, démontré d'importantes capacités de transfert entre langues [K et al., 2020, Wang et al., 2020, Conneau et al., 2020b, Xue et al., 2020]. Étant donné l'intérêt de ces modèles pour le transfert entre langues et par extension pour le LL, il est capital de mieux comprendre ce phénomène ainsi que ses limites.

Dans ce chapitre, nous analysons donc les capacités de transfert entre langues de BERT multilingue et nous travaillons sur la tâche de détection des slots pour les systèmes de dialogue orientés tâche. Afin d'identifier les résultats qui peuvent être

généralisés et ceux qui sont spécifiques à un corpus, nous réalisons aussi nos expériences sur une tâche similaire qui est la reconnaissance d'entité nommées (NER). Ces deux tâches correspondent en effet à un problème d'étiquetage de séquences, où chaque token d'un texte doit être annoté avec une étiquette spécifique. Nous utilisons les corpus multilingues MultiATIS++ [Xu et al., 2020b] pour la tâche de détection des slots, et MultiCoNER pour la tâche de reconnaissance des entités nommées [Malmasi et al., 2022a, Malmasi et al., 2022b] [1].

Un autre aspect du LL qui n'a pas été étudié dans le chapitre précédent correspond à l'apprentissage continu. Ainsi, nous utilisons l'apprentissage continu comme cadre expérimental pour l'étude du transfert entre langues. Nous considérons donc un modèle qui apprend la même tâche au cours du temps mais qui est progressivement adapté sur une séquence de langues. Nous pensons que cette configuration expérimentale est intéressante, non seulement parce qu'elle constitue une nouvelle façon d'étudier le transfert entre langues, mais aussi parce qu'elle est plus adaptée à un cas réel. Comme vu en section 1.1, la capacité d'un système de dialogue à s'adapter à de nouvelles données au cours du temps est une caractéristique hautement recherchée. En effet, la collecte et l'annotation de données sont des étapes coûteuses, ce qui peut induire le fait que des données d'entraînement sont soit indisponibles, soit incomplètes lors du développement d'un système de dialogue. De plus, les exigences du système peuvent évoluer dans le temps en fonction des besoins des utilisateurs. Cela signifie que le modèle doit s'adapter de manière séquentielle à mesure que les données d'entraînement sont disponibles. On peut prendre comme exemple un système de dialogue qui est progressivement déployé dans différents pays et pour lequel les données d'entraînement de chaque langue sont collectées et annotées au fur et à mesure. Cependant, les solutions naïves d'adaptation d'un système existant à une nouvelle langue sont coûteuses, car elles nécessitent soit un nouvel entraînement à chaque nouvelle langue à partir de zéro avec toutes les langues disponibles, soit l'entraînement et le maintien d'un modèle par langue. À l'inverse, entraîner un unique modèle de manière séquentielle sur plusieurs jeux de données qui sont rendus disponibles un par un au cours du temps, est au coeur de l'apprentissage continu. Cependant, si l'apprentissage continu permet en théorie le transfert de connaissances des langues passées vers de nouvelles langues sous la forme de transfert en avant, les modèles d'apprentissage continu sont souvent sujets au problème d'oubli catastrophique, où les performances sur les langues passées diminuent au fur et à mesure de l'apprentissage de nouvelles langues.

1. Nous n'avons pas pu travailler sur le corpus MASSIVE [FitzGerald et al., 2022] étant donné sa récente mise à disposition. Cependant nous supposons que les résultats pourraient être semblables à ceux sur MultiATIS++ étant données les similitudes entre ces deux corpus.

Dans ce chapitre nous cherchons donc à étudier le transfert entre langues dans le cadre de l'apprentissage continu comme illustré en Figure 5.1. Nous nous concentrons sur les questions de recherche suivantes :

— Est-ce qu'un transfert en avant est observable lors de l'apprentissage continu, ou le phénomène d'oubli empêche-t-il le transfert en avant ?
— Quel transfert peut-on observer par rapport à des entraînements monolingues, zero-shot ou multilingues ?
— Quelles sont les capacités que présente un modèle entraîné de manière continue ?

Nous décrivons d'abord en section 5.3 les corpus et le modèle utilisés dans le cadre de nos expériences, puis nous définissons en section 5.4 les différentes métriques utilisées pour les mesures de transfert zero-shot, multilingue et continu. Nous présentons en section 5.5 les mesures de transfert obtenues dans le cadre de l'apprentissage continu sur les corpus MultiATIS++ et MultiCoNER et les comparons aux mesures de transfert zero-shot et multilingues. Ensuite en section 5.6 nous analysons les mesures de transfert continu à différentes positions dans la séquence d'entraînement. Enfin en section 5.7, nous nous intéressons aux capacités d'un modèle entraîné de manière continue, en nous interrogeant sur son aptitude à tendre vers une meilleure initialisation multilingue, et ainsi permettre aux langues touchées par l'oubli de retrouver de bonnes performances.

Les expériences ont été réalisées dans le cadre d'un travail conjoint et équitable avec Juan Manuel Coria, un autre doctorant du LISN. Les expériences et les résultats décrits dans ce chapitre ont été présentés au workshop COLING MMM-PIE (Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models) de 2022 [Coria et al., 2022]. Le code associé à l'article est partagé de manière à permettre la reproduction des expériences [2].

5.2 État de l'art

Apprentissage continu

Comme présenté en section 3.2.2, l'apprentissage continu consiste en un apprentissage qui se fait sur une séquence de tâches différentes. De manière générale, on parle de tâche mais il peut s'agir de l'apprentissage d'une nouvelle classe, d'un nouveau domaine, ou d'une nouvelle tâche du point de vue des sorties attendues. Lors de l'apprentissage d'une nouvelle tâche, l'accès aux données des tâches précédentes n'est en théorie pas autorisé, car cela représente une utilisation linéaire des ressources par rapport à la longueur de la séquence, qui peut être infinie.

2. https://github.com/juanmc2005/ContinualNLU

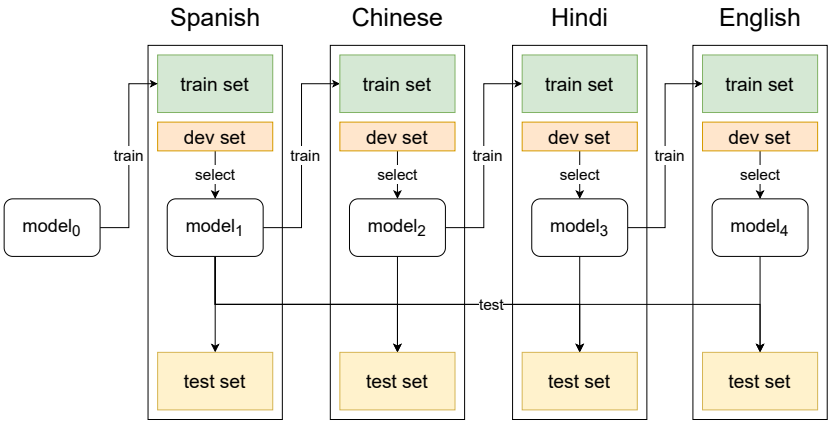


FIGURE 5.1 – Illustration schématique de l'entraînement continu d'un modèle sur une séquence de 4 langues. Pour chaque langue i, le modèle est adapté par fine-tuning sur le jeu de données d'entraînement $train_i$, le meilleur modèle est sélectionné sur le jeu de données de développement dev_i puis le modèle est évalué sur les jeux de données de test de chacune des langues de la séquence.

Notez que l'accès aux données des tâches précédentes est parfois autorisé s'il est limité, comme dans le cas du replay [Robins, 1995]. L'apprentissage continu fait apparaître deux types de transfert : le transfert en avant et le transfert en arrière [Hadsell et al., 2020, Lopez-Paz and Ranzato, 2017, Arora et al., 2019]. Comme vu en section 3.2.2, le transfert en avant désigne le gain de performance sur une tâche pas encore apprise grâce à l'apprentissage continu d'autres tâches. Le transfert en arrière désigne lui le gain de performance sur une tâche déjà apprise grâce à l'apprentissage continu de nouvelles tâches. Cependant, pour la majorité des systèmes d'apprentissage continu, les performances des tâches apprises au début de la séquence ont tendance à diminuer fortement avec l'augmentation du nombre de tâches apprises, résultant en un transfert en arrière négatif. Ce phénomène est connu sous le nom d'oubli catastrophique [Hadsell et al., 2020, French, 1999] et les travaux sur l'apprentissage continu se concentrent généralement sur le développement de méthodes et techniques visant à le diminuer.

À notre connaissance, [Madotto et al., 2020] sont les premiers à avoir appliqué l'apprentissage continu aux systèmes de dialogue en considérant une séquence contenant un nombre important de domaines différents. Ainsi ils évaluent des méthodes d'apprentissage continu connues pour minimiser l'oubli sur une séquence d'apprentissage de 37 domaines dans quatre configurations différentes : la détection des intentions, le suivi de l'état du dialogue, la génération du langage naturel et l'approche bout-en-bout. Les méthodes d'apprentissage continu ont aussi été

utilisées dans un cadre de fine-tuning où les séquences d'apprentissage ne comportent que deux tâches à apprendre. Ainsi, [Lee, 2017] a appliqué une méthode connue de continual learning pour minimiser l'oubli lors de l'adaptation à un nouveau domaine pour leur système de dialogue conversationnel. De leur côté, [Liu et al., 2021b] ont cherché à préserver les performances de modélisation du langage du modèle multilingue BERT (mBERT) à travers la tâche de *Masked Language Modelling*, ainsi que ses capacités de transfert multilingue, lors de l'adaptation de mBERT à une tâche cible en anglais. Les tâches cibles correspondaient à des tâches d'étiquetage de séquences, précisément la tâche de reconnaissance des entités (NER) de CoNLL 2002 [Tjong Kim Sang, 2002] et la tâche de POS-Tagging à l'aide du corpus Universal Depencies 2.0. L'apprentissage continu a aussi été appliqué sur une séquences de classes pour la tâche de NER [Xia et al., 2022, Monaikul et al., 2021]. Comme indiqué dans l'introduction, la tâche de NER nous intéresse puisqu'elle est proche de la tâche de détection des slots nécessaire au fonctionnement des systèmes de dialogue orientés tâche.

Étude du transfert entre langues

La majorité des études de transfert entre langues, appliquées aux tâches de détection des slots et de NER, se concentrent soit sur l'entraînement conjoint des différentes langues, soit sur l'entraînement sur une langue source puis l'application ou le fine-tuning sur une langue cible [Xu et al., 2020b, Schuster et al., 2019, Arkhipov et al., 2019, Mueller et al., 2020, Wang et al., 2020]. Notre étude du transfert se différencie ainsi de ces travaux puisqu'elle se fait dans le cadre de l'apprentissage continu.

De nombreuses études s'intéressent plus largement aux capacités de transfert entre langues de modèles de langue pré-entraînés sur un large corpus multilingue, comme mBERT [Pires et al., 2019, Deshpande et al., 2022].

Dans le cadre de ce chapitre nous nous intéressons donc à l'apprentissage continu d'une séquence de langues sur les tâches de détection de slots et de NER. À notre connaissance ce cadre expérimental n'a jamais été étudié jusqu'ici, l'application de l'apprentissage continu se faisant sur de nouvelles langues et l'étude du transfert se faisant dans un cadre continu.

Corpus	Langue	# Enoncés *train*	# Enoncés *dev*	# Enoncés *test*	# Étiquettes
MultiATIS++	hindi	1 440	160	893	75
	turc	578	60	715	71
	autres	4 488	490	893	84
MultiCoNER	chacune	15,3K	800	≥138K	6

TABLEAU 5.1 – Nombre d'énoncés ou de phrases annotés et nombre d'étiquettes différentes (sans prendre en compte les préfixes `B` et `I`) dans les corpus MultiATIS++ [Xu et al., 2020b] et MultiCoNER [Malmasi et al., 2022a] par jeu de données et pour chaque langue.

5.3 Cadre expérimental général

5.3.1 Corpus de données

Dans le cadre de nos expériences nous nous intéressons à la tâche d'étiquetage de séquences, où chaque token d'une séquence en entrée doit être associé à une étiquette au format IOB (voir Figure 2.3). À noter que pour les deux corpus, les étiquettes à prédire restent les mêmes d'une langue à une autre, de telle sorte que la tâche reste la même au cours de l'apprentissage continu.

MultiATIS++

Le corpus multilingue MultiATIS++ provient du corpus ATIS (Air Travel Information System) [Hemphill et al., 1990b], constitué d'énoncés d'utilisateurs demandant des informations sur des vols en avion. Originellement en anglais (EN), ce corpus se concentre sur les tâches associées au module NLU des systèmes de dialogue orientés tâche, dont fait partie la tâche de détection des slots (voir section 2.2.2). Le corpus MultiATIS++ consiste ainsi en la traduction manuelle du corpus ATIS en six autres langues : l'allemand (DE), le français (FR), l'espagnol (ES), le portugais (PT), le chinois (ZH) et le japonais (JA). Il comprend également deux langues supplémentaires : l'hindi (HI) et le turc (TR), qui ont été ajoutées dans le cadre de MultiATIS [Upadhyay et al., 2018].

Contrairement aux traductions ajoutées dans MultiATIS++, le nombre d'énoncés traduits en hindi et en turc ne sont pas aussi élevés que pour les autres langues (58% et 77% de données en moins respectivement pour l'hindi et le turc). Plus de détails sur la composition de MultiATIS++ sont présentés dans le tableau 5.1.

MultiCoNER

Le corpus MultiCoNER a été créé dans le cadre de la tâche 11 de SemEval 2022 [Malmasi et al., 2022a, Malmasi et al., 2022b] et se concentre sur la tâche de détection des entités nommées (NER). Alors qu'il s'agit habituellement d'une tâche générique consistant à identifier des entités comme des personnes, des organisations, des lieux ou des dates dans des textes écrits, ce corpus propose de se concentrer sur la détection d'entités ambiguës et complexes dans des textes courts et peu contextuels. Ces entités sont des personnes, des lieux, des groupes, des organismes, des produits et des travaux créatifs. MultiCoNER vise également à stimuler la recherche sur les modèles multilingues, puisque il contient des annotations en 11 langues. Pour une comparaison équitable avec MultiATIS++, nos expériences sont réalisées sur le même nombre de langues, soit 9 langues, à savoir le bengali (BN), l'allemand (DE), l'anglais (EN), l'espagnol (ES), l'hindi (HI), le coréen (KO), le néerlandais (NL), le turc (TR) et le chinois (ZH). Plus de détails sur la composition de MultiCoNER sont présentés dans le tableau 5.1. Notez que chaque langue du corpus comporte le même nombre de données pour l'entraînement et la validation mais que le nombre de données pour l'évaluation est variable et très élevé.

Dans le reste du chapitre et pour les deux corpus, nous désignons les jeux de données *train*, *dev* et *test* d'une langue donnée, par un indice i (par exemple $train_i$).

5.3.2 Modèle

Notre modèle repose sur la version de base de BERT multilingue, avec une taille cachée de 768 (177M paramètres) [Devlin et al., 2019b], pré-entraîné sur 104 langues différentes, incluant les langues présentes dans MultiATIS++ et MultiCoNER. Afin de permettre l'étiquetage des tokens, nous avons ajouté un classifieur à propagation avant (feed-forward) de deux couches cachées de taille 768 ainsi qu'une activation ReLU. L'entrée du classificateur est constituée des états cachés du token de la dernière couche après application du dropout avec $p = 0.1$.

Conformément à [Xu et al., 2020b], le modèle est entraîné sur MultiATIS++ avec l'optimiseur Adam, un taux d'apprentissage de 10^{-5} et une taille de batch de 32 pour 50 époques. Le modèle est entraîné sur MultiCoNER de la même manière, avec un taux d'apprentissage de 5×10^{-5} et sur 15 époques (optimisés sur *dev*). Lors de l'entraînement, le modèle avec le meilleur F1-score sur le jeu de données de *dev* est sélectionné. Ce modèle est ensuite évalué sur le jeu de données de *test* associé à l'aide du F1-score. Le F1-score (micro) est calculé avec la librairie `seqeval` [Nakayama, 2018].

5.4 Métriques pour le transfert

Le transfert entre langue peut être défini comme l'amélioration des performances d'un modèle sur une langue grâce à la connaissance d'autres langues. Le transfert peut ainsi prendre plusieurs formes selon le mode d'entraînement du modèle.

Transfert zero-shot

Dans le cas où des données annotées pour la tâche étudiée sont disponibles dans plusieurs langues sauf la langue cible, il est possible d'entraîner un modèle sur les langues de notre choix et d'évaluer le modèle sur la langue cible. On parle alors de transfert zero-shot puisqu'aucune donnée de la langue cible n'est utilisée pendant l'entraînement. La métrique utilisée pour évaluer ce transfert est simplement la performance observée sur la langue cible.

Transfert multilingue

Dans un contexte où toutes les données sont disponibles dès le départ, le transfert peut être envisagé en termes d'entraînement conjoint. Si l'apprentissage conjoint des langues i et j (multilingue) donne de meilleures performances sur j que l'apprentissage sur j uniquement (monolingue), alors il y existe un transfert de connaissances de i à j. Nous définissons ainsi le transfert multilingue pour la langue i comme le gain de performance sur les données de test de la langue i entre le modèle multilingue entraîné sur toute les langues en même temps, y compris la langue i, et le modèle monolingue entraîné seulement sur la langue i, soit :

$$T_i^{\text{multi}} = P_i^{\text{multi}} - P_i^{\text{mono}} \tag{5.1}$$

Transfert continu

Comme vu en section 3.2.2, l'aspect séquentiel de l'apprentissage continu fait apparaître deux sortes de transfert : le transfert en avant, défini comme le gain de performance observé sur une langue pas encore apprise grâce à l'apprentissage d'autres langues ; et le transfert en arrière, défini à l'inverse comme le gain de performance sur une langue déjà apprise grâce à l'apprentissage d'une nouvelle langue. Considérons une séquence de L langues, et la matrice $P^{\text{cont}} \in \mathbb{R}^{L \times L}$ contenant les performances sur les jeux de données de test de chaque langue après avoir appris chaque langue de la séquence, telles que $P_{i,j}^{\text{cont}}$ correspond à la performance sur la langue i après l'apprentissage sur la langue j. Le transfert en avant pour une séquence donnée peut ainsi être défini comme présenté en équation 5.2. Nous nous intéressons particulièrement au transfert en avant sur la dernière langue de

la séquence, afin de comparer le transfert en avant et le transfert multilingue sur le même nombre de langues, comme défini en équation 5.3.

$$T_i^{\text{avant}} = P_{i,i}^{\text{cont}} - P_{i,1}^{\text{cont}} \tag{5.2}$$

$$T_L^{\text{avant}} = P_{L,L}^{\text{cont}} - P_L^{\text{mono}} \tag{5.3}$$

De la même manière, le transfert en arrière pour une séquence donnée peut être défini tel que présenté en équation 5.4. Le transfert en arrière peut être négatif, on parle alors d'oubli, voir d'oubli catastrophique lorsque celui-ci est important. Cependant, cette mesure de transfert peut ne pas être exacte pour certaines langues, puisque $P_{i,i}^{\text{cont}}$ peut avoir bénéficié du transfert en avant. Ainsi, pour isoler la mesure de transfert en arrière de l'effet du transfert en avant, nous mesurons le transfert en arrière seulement sur la première langue de la séquence, tel que défini en équation 5.5.

$$T_i^{\text{arrière}} = P_{i,L}^{\text{cont}} - P_{i,i}^{\text{cont}} \tag{5.4}$$

$$T_1^{\text{arrière}} = P_{1,L}^{\text{cont}} - P_i^{\text{mono}} \tag{5.5}$$

Les mesures de transfert en avant et en arrière définies en équation 5.3 et 5.5 sont différentes de celles que l'on peut trouver dans la littérature. Ces dernières consistent généralement à calculer la moyenne sur l'ensemble des langues des différences de performances à différentes étapes de l'apprentissage continu [Lesort et al., 2019, Lopez-Paz and Ranzato, 2017]. À l'inverse, nos définitions nous permettent d'obtenir des mesures par langue, qui sont comparables avec le transfert multilingue, et qui sont isolées du transfert en avant dans le cas du transfert en arrière.

5.5 Transfert entre langues

Avant d'étudier le transfert entre langues dans le cadre de l'apprentissage continu, nous mesurons le transfert dans différents cadres pour servir de point de comparaison. Dans un premier temps nous nous intéressons au transfert d'un modèle entraîné sur une seule langue lorsqu'il est appliqué sur d'autres langues (*transfert zero-shot*). Puis nous nous intéressons au transfert d'un modèle entraîné conjointement sur l'ensemble des langues disponibles (*transfert multilingue*). Enfin nous mesurons le transfert d'un modèle entraîné sur une séquence de langues (*transfert continu*) et comparons ces résultats avec les mesures de transfert zero-shot et multilingue.

À noter que l'ensemble des expériences présentées ici sont répétées sur 5 valeurs différente de graine aléatoire (*random seed* en anglais) afin de réduire l'effet du

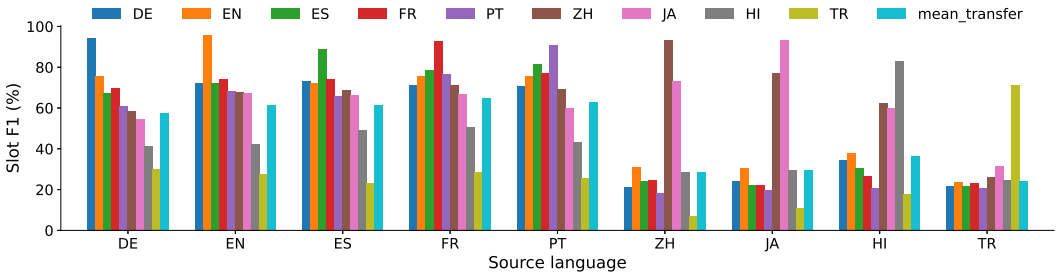


FIGURE 5.2 – Performances obtenues sur chacun des jeux de données $test_i$ de MultiATIS++ après l'entraînement monolingue sur chaque langue source (indiqué en abscisse) moyennées sur 5 entraînements.

hasard [3]. Les performances indiquées dans ce chapitre correspondent donc à la moyenne des performances.

5.5.1 Transfert zero-shot

Configuration des expériences

Pour étudier le transfert zero-shot entre chaque langue, nous examinons les performances entre paires de langues. Nous entraînons le modèle sur une langue source (monolingue), puis nous examinons ses performances sur chacune des autres langues sans entraînement supplémentaire. Pour chaque langue source nous calculons aussi le transfert moyen correspondant à la moyenne des performances du modèle sur toutes les langues sauf la langue source. En particulier, ces expériences permettent d'observer les capacités de transfert entre langues de BERT multilingue.

Résultats

Les résultats des expériences zero-shot sur MultiATIS++ sont présentées dans la figure 5.2. Nous observons deux phénomènes distincts qui sont cohérents avec les études antérieures connexes [Rahimi et al., 2019]. Tout d'abord, le transfert zero-shot semble être maximal pour les langues dont l'écriture est similaire. Par exemple, le chinois atteint ses meilleures performances zero-shot lorsque le modèle est entraîné sur le japonais (et vice-versa), mais les performances des autres langues après un entraînement sur le japonais sont faibles. Cependant, pour le turc nous observons que l'entraînement sur cette langue ne permet qu'un transfert minime

3. Les graines aléatoires ont été fixées aux valeurs 0, 100, 200, 300 et 400 pour permettre la reproductibilité des expériences.

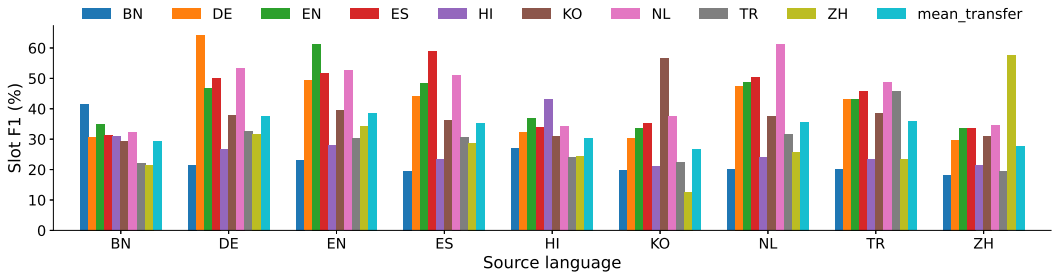


FIGURE 5.3 – Performances obtenues sur chacun des jeux de données $test_i$ de MultiCoNER après l'entraînement monolingue sur chaque langue source (indiqué en abscisse) moyennées sur 5 entraînements.

vers les autres langues qui utilisent aussi un alphabet latin. Cela peut s'expliquer par le fait que le turc dispose d'une quantité de données d'entraînement inférieure aux autres langues (77% de moins). Nous pouvons aussi observer que les paires de langues présentant le meilleur transfert ne sont pas toujours symétriques : la meilleure langue source pour l'espagnol est le portugais, mais pour le portugais, c'est le français. Enfin, nous notons que les performances globales du transfert zero-shot sont les plus élevées lorsque la langue source est européenne, même vers d'autres familles de langues et d'autres écritures.

Les résultats des expériences zero-shot sur MultiCoNER sont présentées dans la figure 5.3. Les modèles entraînés sur des langues européennes (en l'occurrence l'allemand, l'anglais, l'espagnol et le néerlandais) présentent des transferts similaires, ainsi que le coréen et le chinois, de la même manière que l'hindi et le bengali. De manière générale nous observons donc que les transferts sont similaires pour les langues d'une même famille géographique. Cependant le fait d'appartenir à la même famille de langue ne semble pas toujours être profitable en termes de transfert zero-shot. Par exemple, le chinois est la langue qui profite le moins du transfert lorsque la langue source est le coréen. Étonnamment, le néerlandais semble profiter du transfert de manière significative pour n'importe quelle langue source, et ses performances zero-shot sont même supérieures aux performances du turc alors que la langue source est le turc. Puisque les jeux de données ne sont pas parallèles entre langue, nous pouvons nous demander si le jeu de données de test du néerlandais ne serait pas plus simple que les jeux de données des autres langues. Les modèles entraînés sur le turc présentent aussi des transferts similaires vers les langues européennes, ce qui peut s'expliquer par le fait que le turc partage le même alphabet.

Pour essayer de se représenter les capacités de transfert de chaque langue sources, nous calculons le transfert moyen de chaque langue source vers les autres langues.

Nous observons que la langue source qui présente le transfert moyen le plus bas est le coréen, avec un transfert moyen de 26,61 de F1, tandis que celle qui présente le transfert moyen le plus haut est l'anglais avec un transfert moyen de 38,60 de F1.

5.5.2 Transfert multilingue

Configuration des expériences

Afin de mesurer le transfert dans un cas où toutes les données de chaque langue sont disponibles dès le début, nous entraînons le modèle sur toutes les langues de manière conjointe (multilingue). Nous comparons ensuite les performances multilingues avec celles obtenues avec un entraînement monolingue afin d'obtenir les mesures T_i^{multi} comme défini en équation 5.1. À noter que l'entraînement multilingue se fait sur la concaténation de tous les jeux de données d'entraînement $train_i$ et que le meilleur modèle est sélectionné à l'aide de la concaténation de tous les jeux de données de validation dev_i. Le modèle est ensuite évalué sur chaque jeu de données $test_i$ et nous rapportons pour chaque langue la moyenne et l'écart type des valeurs de F1-score sur ces jeux de données. Le transfert multilingue T_i^{multi} (voir équation 5.1) rapporté dans cette section correspond à la différence des moyennes des performances multilingues et monolingues.

Résultats

Les résultats sur MultiATIS++ sont présentés dans le tableau 5.2. Nous observons que les performances multilingues sont plus élevées que les performances monolingues (sauf pour le chinois et le japonais), ce qui confirme l'existence d'un transfert entre langues multilingue. Pour les langues européennes (allemand, anglais, espagnol, français et portugais) le transfert est modeste mais visible, tandis que les langues asiatiques (chinois et japonais) ne semblent pas bénéficier du transfert. Pour les deux langues à faibles ressources que sont l'hindi et le turc, le transfert est remarquable avec respectivement une amélioration de 4,8% et 13,9%. Comme souligné dans [Do et al., 2020], les traductions MultiATIS++ conservent les mêmes valeurs de slots : par exemple, dans des énoncés du turc ne sont indiqués que des noms de ville américaines pour le type de slot destination city. Nous pensons que cela peut expliquer pourquoi le transfert est important ici. Le fait que le corpus contienne moins de données d'entraînement pour l'hindi et le turc que pour les autres langues pourrait également expliquer pourquoi le transfert multilingue est beaucoup plus élevé pour ces deux langues. De plus, comme les performances monolingues sont déjà hautes pour toutes les langues, sauf les langues à faibles ressources et que seules ces dernières langues semblent bénéficier de manière significative du transfert, nous pouvons supposer que la tâche de détection des slots sur

MultiATIS++ est facilement résolue à l'aide notre modèle étant donné le nombre de données disponibles par défaut. Ces observations rejoignent celles faites sur le corpus ATIS [Béchet and Raymond, 2018]. Le corpus MultiATIS++ semble cependant intéressant s'il est appliqué à des langues à faibles ressources, à l'image de l'hindi et du turc.

Le tableau 5.3 montre les résultats sur MultiCoNER. Les résultats monolingues sont beaucoup plus faibles que dans MultiATIS++ même si le nombre d'étiquettes à prédire est bien inférieur, ce qui suggère que MultiCoNER est une tâche plus complexe que MultiATIS++. Bien que le corpus ne soit pas parallèle, nous observons un transfert multilingue important pour l'ensemble des langues [4]. Ceci est surprenant lorsque l'on sait que seulement un maximum de 8% des mentions d'entités, qui apparaissent dans le jeu de données de test d'une langue, sont communes à celles qui apparaissent dans le jeu de données d'entraînement des autres langues. Si nous comparons les résultats sur MultiATIS++ et MultiCoNER et la quantité de données disponibles pour chaque corpus, cela suggère que la tâche associée à MultiCoNER nécessite beaucoup plus de connaissances pour être résolue que celle de MultiATIS++. Cela nous amène à penser de nouveau que la tâche associée à MultiCoNER est plus compliquée que celle de MultiATIS++. Ainsi, contrairement à MultiATIS++, où seules les langues à faibles ressources bénéficiaient d'un transfert de plus de 1,0 points de F1-score, la majorité des langues de MultiCoNER profitent d'un transfert multilingue supérieur à ce même nombre.

Cependant, l'entraînement multilingue suppose que les données de toutes les langues sont disponibles en même temps. Comme nous l'avons déjà mentionné, ce n'est que rarement le cas en pratique, car la collecte et l'annotation de données sont coûteuses. De plus, étant donné N le nombre maximum d'énoncés par langue et L le nombre de langues, l'entraînement sur une nouvelle langue a un coût en temps $O(LN)$, puisque le modèle entier doit être entraîné à partir de zéro à chaque fois. Une solution naïve consiste à utiliser plusieurs modèles monolingues, ce qui augmente cependant le coût de stockage en mémoire à $O(LN)$. La réduction des deux coûts à $O(N)$ motive notre décision de structurer l'entraînement comme une séquence.

5.5.3 Transfert continu

Configuration des expériences

Étant donné une séquence d'apprentissage (une liste de langues dans un ordre donné), l'apprentissage continu consiste à entraîner le modèle sur $train_i$ et à le

4. Sauf pour le chinois où le transfert multilingue est négligeable si l'on considère l'écart type.

valider sur dev_i pour chaque langue i dans l'ordre donné, comme le montre la figure 5.1. Bien que cet apprentissage ne nécessite pas l'accès aux données de l'ensemble des langues et que son adaptation à de nouvelles langues soit l'option la moins coûteuse, cette approche peut être sujette au phénomène d'oubli sur les langues précédemment apprises. Dans cette section, nous rapportons les mesures de transfert en avant et en arrière (inverse de l'oubli) pour chaque langue et sur les deux corpus.

Comme indiqué en section 5.4, pour le transfert en avant nous nous intéressons seulement aux performances finales de la *dernière* langue de la séquence P_{LL}^{cont} (*e.g.* modèle$_4$ évalué sur l'anglais dans la Figure 5.1), tandis que pour le transfert en arrière nous nous intéressons seulement aux performances finales de la *première* langue P_{1L}^{cont} (*e.g.* modèle$_4$ évalué sur l'espagnol dans la Figure 5.1). Ainsi, durant les expériences continues, nous entraînons le modèle sur plusieurs séquences de langues, afin de faire apparaître chaque langue autant de fois au début et à la fin d'une séquence. Ceci permet d'obtenir des mesures de transfert en avant et en arrière pour chacune des langues. Plus précisément, pour chaque langue, chaque type de transfert et chaque corpus, trois séquences d'entraînement différentes sont définies, résultant en un total de 108 séquences [5]. Les entraînement sont ensuite répétées sur cinq graines aléatoires, ce qui représente un ensemble de 540 expériences. Les trois séquences par langue sont choisies aléatoirement de manière à maximiser le coefficient de corrélation de rang de Kendall [Abdi, 2007]. Ce coefficient est utilisé comme critère de distance afin d'obtenir des séquences de langues les plus différentes que possible les unes des autres. Les performances par langue P_{LL}^{cont} et P_{1L}^{cont} sont moyennées sur les 5 graines aléatoires et sur les 3 séquences afin d'obtenir des mesures de transfert qui ne soient pas trop dépendantes de l'ordre des langues. Les mesures de transfert par langue T_i^{multi} (équation 5.1), T_L^{avant} (équation 5.3) et $T_1^{\text{arrière}}$ (équation 5.5) correspondent donc aux différences des moyennes des performances.

Résultats

Nous cherchons d'abord à savoir si l'entraînement continu permet le transfert de connaissances vers une langue encore jamais rencontrée pendant l'entraînement. En d'autres mots, nous cherchons à savoir si un transfert en avant est présent en examinant les performances moyennes P_{LL}^{cont} et en les comparant aux performances monolingues et multilingues. En parallèle, nous vérifions si le modèle « oublie » les langues acquises précédemment, au fur et à mesure de l'apprentissage continu d'autres langues. Pour cela nous examinons le transfert en arrière en comparant les performances moyennes P_{1L}^{cont} aux performances monolingues.

5. Neuf langues, trois séquences, deux types de transfert et deux corpus.

Training	DE	EN	ES	FR	PT	ZH	JA	HI	TR
Monolingual	94.4 (0.2)	95.6 (0.1)	88.9 (0.4)	93.2 (0.1)	90.3 (0.6)	93.3 (0.4)	93.1 (0.4)	82.4 (0.5)	71.3 (0.9)
Multilingual	95.0 (0.2)	96.0 (0.2)	90.4 (0.4)	94.0 (0.3)	91.4 (0.2)	93.6 (0.2)	93.0 (0.1)	87.2 (0.3)	85.2 (0.6)
T_i^{multi}	+0.6	+0.4	+1.5	+0.8	+1.1	+0.3	-0.1	+4.8	+13.9
Continual (P_{LL}^{cont})	94.9 (0.2)	95.9 (0.1)	89.9 (0.5)	93.9 (0.3)	91.3 (0.3)	93.9 (0.3)	93.1 (0.3)	85.6 (0.7)	84.0 (0.6)
T_L^{avant}	+0.5	+0.3	+1.0	+0.7	+1.0	+0.6	+0.0	+3.2	+12.7
Continual (P_{1L}^{cont})	94.0 (0.7)	95.5 (0.2)	89.2 (0.5)	91.4 (1.7)	88.4 (4.9)	92.0 (1.0)	91.7 (0.7)	80.5 (1.8)	68.1 (3.5)
$T_1^{\text{arrière}}$	-0.4	-0.1	+0.3	-1.8	-1.9	-1.3	-1.4	-1.9	-3.2

TABLEAU 5.2 – Mesures de F1-score sur la tâche de détection des slots de MultiATIS++ sur $test_i$ pour les expériences monolingues, multilingues et continues, accompagnées des mesures de transfert correspondantes. P_{LL}^{cont} indique le F1-score de la langue indiquée en colonne, à la fin de l'apprentissage continu, moyenné sur les 3 séquences d'apprentissage où cette même langue est situé en *dernière* position. P_{1L}^{cont} indique le F1-score de la langue indiquée en colonne, à la fin de l'apprentissage continu, moyenné sur les 3 séquences d'apprentissage où cette même langue est situé en *première* position. Les valeurs de F1-scores sont aussi moyennées sur 5 expériences et l'écart type est donné entre parenthèses.

Training	BN	DE	EN	ES	HI	KO	NL	TR	ZH
Monolingual	41.6 (3.2)	64.1 (0.8)	61.3 (0.6)	59.0 (0.8)	43.1 (1.2)	56.7 (0.7)	61.4 (0.9)	45.7 (0.7)	57.6 (0.8)
Multilingual	44.9 (1.6)	66.9 (0.4)	64.4 (0.7)	63.8 (0.4)	46.4 (1.2)	59.4 (0.8)	66.5 (0.5)	50.6 (1.0)	58.2 (1.0)
T_i^{multi}	+3.3	+2.8	+3.1	+4.8	+3.3	+2.7	+5.1	+4.9	+0.6
Continual (P_{LL}^{cont})	43.4 (1.8)	66.0 (0.6)	63.0 (0.6)	62.1 (0.9)	44.2 (1.0)	57.0 (0.7)	64.6 (0.6)	50.1 (0.8)	56.2 (1.3)
T_L^{avant}	+1.8	+1.9	+1.7	+3.1	+1.1	+0.3	+3.2	+4.4	-1.4
Continual (P_{1L}^{cont})	31.7 (4.5)	50.9 (1.5)	52.5 (2.6)	51.1 (2.3)	32.2 (2.4)	43.2 (2.4)	55.4 (3.4)	37.4 (1.9)	40.0 (2.8)
$T_1^{\text{arrière}}$	-9.9	-13.2	-8.8	-7.9	-10.9	-13.6	-6.0	-8.3	-17.6

TABLEAU 5.3 – Mesures de F1-score sur la tâche de reconnaissance des entités nommées de MultiCONER sur $test_i$ pour les expériences monolingues, multilingues et continues, accompagnées des mesures de transfert correspondantes. Les informations supplémentaires données dans le tableau 5.2 s'appliquent ici aussi.

Les résultats sur MultiATIS++ sont présentés dans le tableau 5.2. Nous observons que l'entraînement continue bénéficie du transfert entre langues puisque les mesures de transfert T_L^{avant} sont toutes positives. Cependant le transfert en avant continu est toujours inférieur au transfert multilingue [6]. Pour les langues européennes, de manière similaire à un entraînement multilingue, l'entraînement continu bénéficie d'un transfert en avant faible mais significatif. Les langues à faibles ressources que sont l'hindi et le turc bénéficient quant à elles d'un transfert en avant important. Cependant sur ces langues l'écart entre transfert en avant et transfert multilingue se creuse légèrement puisque le transfert en avant perd

6. Sauf pour le chinois et le japonais, mais l'écart n'est pas significatif si nous considèrons les écart types.

Corpus	Training	Model Cost		Data Cost
		Time	Space	Space
MultiATIS++	Monolingual	≤224K	1.6B	≤4K
	Multilingual	1.7M	178M	33K
	Continual	≤224K	178M	≤4K
MultiCoNER	Monolingual	765K	1.6B	15K
	Multilingual	6.9M	178M	138K
	Continual	765K	178M	15K

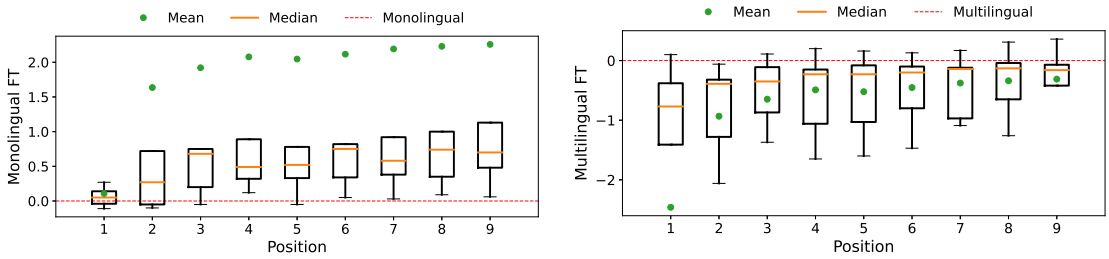
TABLEAU 5.4 – Description de différents types de coût liés à l'ajout d'une nouvelle langue selon les types d'entraînement pour les corpus MultiATIS++ et MultiCoNER. « Model Time Cost » indique le coût lié à l'ajout d'une nouvelle langue en termes d'itération durant l'entraînement. « Model space Cost » indique le coût lié la taille du ou des modèle(s) à garder en mémoire en termes de nombre de paramètres. « Data Cost » indique le coût lié au nombre maximum de données d'entraînement à garder en mémoire en même temps.

respectivement 1,6 et 1,2 points de F1-score par rapport au transfert multilingue.

En parallèle, nous observons que les mesures de transfert en arrière sont presque toutes négatives ; les performances finales moyennes de la première langue P_{1L}^{cont} descendent sous les performances monolingues. La seule exception est l'espagnol, qui bénéficie d'un faible transfert en arrière positif. Cependant, même si le transfert en arrière est négatif, il n'est pas significatif si l'on considère les écarts types, de telle maniière que l'on ne peut pas parler d'oubli catastrophique. Les écarts types sont d'ailleurs plus importants pour P_{1L}^{cont} que pour P_{LL}^{cont}. Nous supposons que le transfert en arrière est plus sensible à l'ordre de la séquence d'entraînement que le transfert en avant, notamment parce que le fait de finir l'entraînement par la langue cible permet de stabiliser les performances sur la langue cible.

Les résultats sur MultiCoNER sont présentés dans le tableau 5.3. De la même manière que pour MultiATIS++, nous observons la présence d'un transfert en avant mais celui-ci reste inférieur au transfert multilingue. De plus, bien que le transfert vers l'avant soit élevé en général, il n'est pas significatif pour le bengali, l'hindi et le coréen étant donné qu'il est inférieur à l'écart type, et est même négatif pour le chinois. Comme pour MultiATIS++, le transfert en arrière est négatif. Cependant, contrairement à ce dernier, l'oubli sur MultiCoNER est bien plus important et est supérieur à l'écart type, si bien que l'on peut parler ici d'oubli catastrophique sur la première langue de la séquence.

Dans l'ensemble, nous pouvons constater que l'entraînement continu bénéficie du transfert vers l'avant aussi bien sur MultiATIS++ que sur MultiCoNER, bien



(a) $P_{ii}^{\text{cont}} - P_i^{\text{mono}}$ (b) $P_{ii}^{\text{cont}} - P_i^{\text{multi}}$

FIGURE 5.4 – Distribution sur la position i de la séquence, de la différence entre la performance continue de la langue à la position i après l'apprentissage ce cette même langue avec la performance monolingue (a) et la performance multilingue (b). (a) est égal à T_i^{avant} (voir équation 5.2). Les valeurs aberrantes ne sont pas visibles pour des questions de lisibilité.

qu'il ne soit toujours pas aussi performant que le modèle multilingue. Cependant l'entraînement continu est impacté de manière claire par le phénomène d'oubli, allant même vers de l'oubli catastrophique pour MultiCoNER.

5.6 Analyses sur la séquence d'entraînement

Afin de mieux comprendre l'effet de la séquence d'entraînement sur le transfert, nous examinons d'abord le transfert en avant à chaque position de la séquence par rapport aux perfomances monolingues et multilingues. Dans un deuxième temps, nous étudions l'impact de la longueur de la séquence d'entraînement sur le transfert en arrière mesuré sur la première langue. Ces analyses sont menées uniquement sur MultiATIS++ en raison de contraintes de temps et de calcul [7]. Dans les figures de cette section, la moyenne, la médiane et les percentiles prennent en compte les éventuelles valeurs aberrantes (*outliers*), alors que le minimum et le maximum ne le font pas.

5.6.1 Transfert en avant par position

Nous cherchons ici à mesurer le transfert à différentes positions de la séquence d'entraînement. Les P_{ii}^{cont} ont d'abord été moyennées sur l'ensemble des expé-

7. Les jeux de données d'évaluation sur MultiCoNER comptant un nombre important de données, l'évaluation est gourmande en termes de ressources de temps et de calcul. Ainsi les évaluations pour les expériences continues sur MultiCoNER n'ont eu lieu qu'à la fin de la séquence et non après l'apprentissage de chaque langue. Par conséquent les analyses décrites ici ne sont pas réalisables sur MultiCoNER.

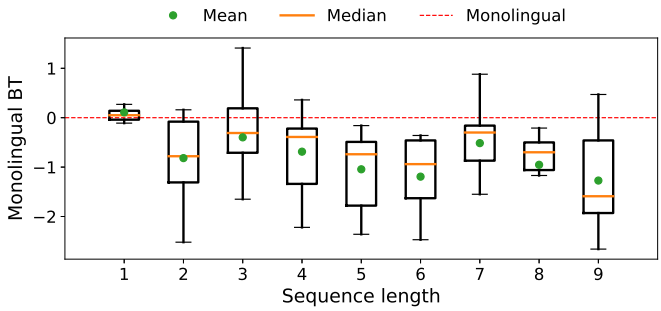


FIGURE 5.5 – Mesures de transfert en arrière $T_1^{\text{arr}} = P_1^{\text{mono}} - P_{1j}^{\text{cont}}$ pour différentes longueurs de séquence j en abscisses.

riences continues réalisées (54 séquences) par position i et par langue, afin qu'une langue n'aie pas plus d'impact qu'une autre. Ensuite la différence entre les P_{ii}^{cont} de chaque langue et de chaque position a été calculée par rapport aux performances monolingues P_i^{mono} et multilingues P_i^{multi} de la langue correspondante. Enfin, les différences ont été moyennées sur l'ensemble des langues comme représenté par les diagrammes en boîte de la Figure 5.4b. Nous observons qu'à l'exception de la première position, où les performances sont équivalentes aux performances monolingues [8], le modèle bénéficie systématiquement du transfert à tout moment de la séquence, puisque les performances continues sont supérieures aux performances monolingues. De plus, si nous pouvons observer globalement que le transfert en avant est plus élevé pour les langues sont situées vers la fin de la séquence, il semble que les gains en matière de transfert en avant sont les plus élevés en début de séquence. Il est intéressant de noter qu'en raison de certaines valeurs aberrantes, généralement l'hindi et le turc, les moyennes sont de mauvaises estimations de la distribution lorsque l'on mesure le transfert en avant. Ceci indique que les mesures de transfert continu couramment utilisées peuvent surestimer ou sous-estimer les performances réelles lorsque le transfert n'est pas uniformément distribué entre les langues. En effet, ces mesures consistent généralement en des moyennes sur l'axe d'adaptation [Lopez-Paz and Ranzato, 2017]. Dans la figure 5.4b, nous observons également que les performances continues se rapprochent des performances multilingues plus on s'approche de la fin de la séquence.

5.6.2 Impact de la longueur de la séquence sur le transfert en arrière

Le transfert en arrière à différentes positions de la séquence n'est pas observable puisqu'il n'est pas possible d'isoler le transfert en arrière du transfert en avant pour les langues qui ne sont pas situées en première position dans la séquence. Ainsi nous nous intéressons à l'impact de la longueur de la séquence sur le transfert en arrière de la première langue de la séquence. Pour ce faire, nous avons moyenné les performances continues de la première langue tout au long de l'entraînement P_{1j}^{cont} sur l'ensemble des expériences continues (54 séquences) par langue et par longueur de séquence j. Nous avons ensuite calculé la différence entre les performances P_{1j}^{cont} et les performances monolingues de la langue correspondante P_1^{mono}. Enfin, les différences ont été moyennées sur l'ensemble des langues comme représenté par le diagramme en boîte de la Figure 5.5. Nous observons que les performances de la première langue sont en général moins bonnes que les performances monolingues, quelque soit la longueur de la séquence considérée. Nous observons par ailleurs que la perte de performance n'est pas strictement monotone, ce qui signifie que la mesure de l'oubli entre le début et la fin de la séquence peut ne pas être suffisante pour expliquer comment le modèle oublie. Notez qu'une séquence de longueur $L = 7$ aurait montré moins d'oubli qu'une séquence de longueur $L = 5$.

5.6.3 Transfert en arrière dans le cadre du curriculum learning

Nous cherchons ici à analyser l'impact de l'ordre des langues dans la séquence d'apprentissage continu sur le transfert en arrière. Plus précisément, nous nous demandons si le choix de la prochaine langue à apprendre, étant donné des critères de difficulté, pourrait impacter de manière positive le transfert en arrière (c'est-à-dire diminuer l'oubli). À noter que l'on s'éloigne ici légèrement du contexte expérimental décrit en début de chapitre puisque l'on définit à l'avance la séquence d'apprentissage dans un cadre proche du curriculum learning (voir section 3.2.7).

Nous générons d'abord 2 séquences d'apprentissage par langue que l'on nomme « easy first » et « hard first » comme présenté dans l'algorithme 2. Pour chaque langue, le processus de génération est répété avec 5 valeurs de graine aléatoire différentes. Pour les séquences « easy first » et « hard first » nous choisissons à chaque position la langue qui a été la plus choisie parmi les 5 séquences générées. L'ensemble des séquences finales est listé dans le tableau 5.5.

8. P_i^{mono} correspond à la moyenne du F1-score sur 5 graines aléatoires donc les performances ne sont pas strictement égales.

« easy first »									« hard first »								
1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
DE	TR	HI	ES	ZH	PT	JA	FR	EN	DE	EN	FR	PT	ES	ZH	JA	HI	TR
EN	TR	HI	ES	JA	PT	DE	ZH	FR	EN	FR	PT	DE	ZH	JA	ES	HI	TR
ES	TR	HI	JA	PT	DE	ZH	FR	EN	ES	FR	EN	PT	DE	ZH	JA	HI	TR
FR	TR	HI	ES	JA	DE	ZH	PT	EN	FR	EN	PT	DE	ES	ZH	JA	HI	TR
PT	TR	HI	ZH	ES	DE	JA	FR	EN	PT	FR	EN	DE	ES	ZH	JA	HI	TR
ZH	TR	HI	PT	DE	ES	JA	FR	EN	ZH	JA	EN	FR	PT	DE	ES	HI	TR
JA	TR	HI	PT	ES	DE	ZH	FR	EN	JA	ZH	EN	FR	PT	DE	ES	HI	TR
HI	TR	PT	ZH	ES	DE	JA	FR	EN	HI	ZH	JA	EN	FR	PT	DE	ES	TR
TR	ES	HI	PT	JA	DE	ZH	FR	EN	TR	JA	ZH	EN	DE	FR	PT	ES	HI

TABLEAU 5.5 – Séquences générées grâce à l'algorithme 2 sur MultiATIS++.

Séquence	**Mesure**	**DE**	**EN**	**ES**	**FR**	**PT**	**ZH**	**JA**	**HI**	**TR**
Kendall	P_{1L}^{cont}	94.0 (0.7)	95.5 (0.2)	89.2 (0.5)	91.4 (1.7)	88.4 (4.9)	92.0 (1.0)	91.7 (0.7)	80.5 (1.8)	68.1 (3.5)
(3 seq./lang.)	$T_1^{\text{arrière}}$	-0.4	-0.1	+0.3	-1.8	-1.9	-1.3	-1.4	-1.9	-3.2
« easy first »	P_{1L}^{cont}	93.0 (0.9)	94.8 (0.2)	87.8 (0.7)	90.9 (0.8)	88.8 (0.6)	91.2 (0.6)	91.6 (0.5)	82.3 (0.8)	71.7 (2.8)
(1 seq./lang.)	$T_1^{\text{arrière}}$	-1.4	-0.8	-1.1	-2.3	-1.5	-2.1	-1.5	-0.1	+0.4
« hard first »	P_{1L}^{cont}	93.8 (1.1)	95.7 (0.2)	90.0 (0.4)	92.9 (0.5)	90.4 (0.4)	92.5 (0.4)	92.3 (0.7)	80.8 (1.3)	66.6 (3.3)
(1 seq./lang.)	$T_1^{\text{arrière}}$	-0.6	+0.1	+1.1	-0.3	+0.1	-0.8	-0.8	-1.6	-4.7

TABLEAU 5.6 – Transfert en arrière $T_1^{\text{arrière}}$ (voir équation 5.5) pour différents types de séquences.

Après la génération de l'ensemble des 2 séquences par langues, les expériences continues sont lancées sur ces séquences et, comme précédemment, chaque expérience est répétée sur 5 graines aléatoires différentes. Les résultats sont donnés dans le tableau 5.6. Quand la première langue de la séquence est l'hindi ou le turc, il est plus profitable que la prochaine langue à apprendre soit la plus facile, plutôt que la plus difficile, pour obtenir un oubli plus faible ; Pour ces deux langues, la dernière langue apprise dans la séquence est respectivement le turc et l'hindi. Comme le transfert zero-shot entre ces deux langues est faible, nous pouvons supposer que c'est l'ordre de la séquence seule qui permet d'avoir un oubli plus faible. À l'inverse, pour les langues européennes et asiatiques, l'oubli est systématiquement plus important pour les séquences « easy first ». Pour ces séquences, les deux dernières langues apprises sont toujours l'hindi et le turc, donc nous pouvons supposer que finir l'apprentissage continu par ces deux langues augmente l'oubli, puisqu'elles sont plus éloignées des langues européennes et asiatiques.

Algorithm 2 Génération des séquences d'apprentissage pour l'analyse de l'impact sur le transfert en arrière de l'ordre des langues dans la séquence selon un critère de difficulté.

Input : the model initialized, the difficulty of the next language to learn (easy or hard), the set of languages

Output : list of generated sequences

```
1: procedure GENERATE_SEQUENCES(model, difficulty, languages)
2:     initialize sequences as an empty list of sequences
3:     for first_lang in languages do
4:         initialize s as an ordered list with first_lang as first element
5:         initialize remaining with languages from which first_lang is removed
6:         while remaining is not empty do
7:             l = last element of s
8:             train model on l
9:             evaluate model on languages from remaining
10:            if difficulty = 'easy' then
11:                set next_l as the language with best evaluation results
12:            else
13:                set next_l as the language with worst evaluation results
14:            add next_l to s
15:            remove next_l from remaining
16:        add s to sequences
17:    return sequences
```

5.7 Capacités d'un modèle entraîné de manière continue

Dans les expériences continues des tableaux 5.2 et 5.3 ainsi que dans les analyses illustrées dans les Figures 5.4 et 5.5, nous observons que le transfert en arrière se détériore au fur et à mesure que l'on s'approche de la fin de la séquence, alors que le transfert vers l'avant s'améliore. Puisque le transfert en arrière négatif, aussi appelé oubli, tend à être lié à une perte de connaissances précédemment acquises, il est surprenant que les performances sur les langues inconnues continuent à augmenter tandis que les performances des langues connues diminuent. Nos résultats indiquent que les connaissances préservées qui facilitent l'acquisition d'une nouvelle langue par BERT multilingue pour la tâche d'étiquetage de séquences ne sont pas les mêmes que celles qui préservent les performances des langues précédentes. Cela pourrait s'expliquer par une transition progressive des paramètres du modèle vers une meilleure initialisation multilingue, spécifique aux corpus d'entraînement, qui pourrait cependant ne pas conserver les spécificités des langues précédentes.

Cette hypothèse motive les expériences complémentaires de cette section qui s'intéressent ainsi aux capacités que possède un modèle entraîné de manière continue. Dans un premier temps nous cherchons à vérifier si les modèles continus tendent en effet vers une meilleure initialisation multilingue. Par la suite nous examinons si les performances initiales de la première langues qui ont été impactées par le phénomène d'oubli peuvent être récupérée rapidement à partir d'un modèle continu. Cette capacité de récupération serait particulièrement intéressante pour le corpus MultiCoNER pour lequel le phénomène d'oubli est fortement présent. Les expériences sont menées sur les deux corpus MultiATIS++ et MultiCoNER.

5.7.1 Vers une meilleure initialisation multilingue

Dans un premier temps nous comparons les performances multilingues du modèle initial model_0 (composé de BERT multilingue et d'un classificateur aléatoire) à celles de model_L, le modèle à la fin de la séquence d'entraînement continu (*e.g.* modèle_4 dans la Figure 5.1). En particulier, nous entraînons [9] les deux modèles sur toutes les langues conjointement pour différents nombres d'époques et nous les évaluons sur chaque langue (entraînement multilingue « multi. »). Notez que model_L provient de nos expériences continues P_{1L}^{cont} et que les performances de ce modèle sont moyennées sur les 27 séquences (voir tableaux 5.2 et 5.3). Les performances de model_0 sont quant à elles moyennées sur 5 graines aléatoires. Les résultats sont présentés dans les tableaux 5.7 et 5.8.

9. Pour les modèles continus model_L il s'agit plus précisément d'un fine-tuning.

Model	Epochs	DE	EN	ES	FR	PT	ZH	JA	HI	TR
$model_0$ multi.	1	82.7 (1.2)	83.6 (0.7)	78.2 (0.3)	80.7 (0.7)	79.4 (0.5)	83.5 (0.7)	82.7 (1.0)	79.6 (0.7)	69.8 (1.5)
	5	94.7 (0.2)	95.3 (0.2)	89.9 (0.2)	93.2 (0.2)	90.7 (0.2)	94.0 (0.2)	93.2 (0.5)	85.9 (0.3)	83.6 (0.7)
	50	95.0 (0.2)	96.0 (0.2)	90.4 (0.4)	94.0 (0.3)	91.4 (0.2)	93.6 (0.2)	93.0 (0.1)	87.2 (0.3)	85.2 (0.6)
$model_L$ multi.	1	94.8 (0.3)	95.9 (0.2)	89.7 (0.6)	93.8 (0.3)	91.2 (0.4)	93.6 (0.5)	93.3 (0.3)	85.7 (0.9)	82.8 (1.3)
	5	94.9 (0.2)	95.9 (0.2)	90.0 (0.5)	93.9 (0.3)	91.3 (0.4)	93.7 (0.4)	93.3 (0.3)	86.0 (0.8)	83.4 (1.0)
$model_L$ + rnd clf multi.	1	93.1 (0.5)	93.7 (0.5)	87.9 (0.5)	91.1 (0.5)	88.5 (0.6)	92.6 (0.5)	92.3 (0.6)	83.4 (0.8)	80.8 (1.3)
	5	94.8 (0.2)	95.8 (0.2)	89.9 (0.5)	93.6 (0.3)	91.1 (0.4)	93.7 (0.4)	93.3 (0.3)	86.3 (0.6)	84.1 (0.8)
$model_0$ mono.	50	94.4 (0.2)	95.6 (0.1)	88.9 (0.4)	93.2 (0.1)	90.3 (0.6)	93.3 (0.4)	93.1 (0.4)	82.4 (0.5)	71.3 (0.9)
$model_L$ mono.	1	95.1 (0.2)	95.8 (0.2)	90.2 (0.4)	93.6 (0.4)	91.2 (0.4)	93.5 (0.5)	93.4 (0.2)	86.3 (0.6)	79.1 (1.5)
	5	95.0 (0.2)	95.8 (0.2)	90.0 (0.4)	94.0 (0.2)	91.3 (0.2)	93.8 (0.4)	93.4 (0.2)	86.7 (0.4)	81.6 (0.8)
	10	95.1 (0.2)	95.8 (0.2)	90.0 (0.5)	93.9 (0.3)	91.3 (0.4)	93.8 (0.4)	93.4 (0.2)	86.7 (0.4)	82.2 (0.9)

TABLEAU 5.7 – Mesures de F1-score sur les différents jeux de données d'évaluation $test_i$ pour les expériences sur l'étude des capacités des modèles continus sur MultiATIS++. $model_0$ mono/multi correspond au modèle initial entraîné sur les données monolingues ou multilingues. $model_L$ correspond au modèle obtenu à la fin de l'entraînement continu. L'écart type est donné entre parenthèses.

Model	Epochs	BN	DE	EN	ES	HI	KO	NL	TR	ZH
$model_0$ multi.	1	36.2 (1.4)	63.1 (0.8)	61.6 (0.6)	60.5 (0.6)	40.5 (1.4)	56.9 (0.4)	63.5 (0.7)	45.5 (0.6)	53.1 (2.4)
	5	43.0 (1.1)	66.6 (1.0)	63.9 (0.2)	63.7 (0.6)	45.4 (1.5)	58.9 (0.7)	66.3 (0.7)	49.7 (1.4)	57.7 (1.5)
	15	44.9 (1.6)	66.9 (0.4)	64.4 (0.7)	63.8 (0.4)	46.4 (1.2)	59.4 (0.8)	66.5 (0.5)	50.6 (1.0)	58.2 (1.0)
$model_L$ multi.	1	42.7 (1.7)	65.8 (0.7)	63.6 (0.7)	63.0 (0.8)	44.8 (1.4)	58.8 (1.0)	65.9 (0.8)	49.8 (1.0)	56.7 (1.3)
	5	43.8 (1.4)	66.4 (0.6)	64.1 (0.5)	63.5 (0.6)	45.4 (1.1)	59.2 (0.8)	66.4 (0.5)	50.6 (0.9)	57.6 (1.2)
$model_L$ + rnd clf multi.	1	42.6 (1.8)	65.5 (0.7)	63.3 (0.6)	62.7 (0.8)	44.7 (1.3)	58.7 (0.8)	65.7 (0.7)	49.6 (1.2)	56.6 (1.4)
	5	43.7 (1.4)	66.3 (0.6)	63.9 (0.6)	63.4 (0.7)	45.2 (1.1)	59.1 (0.8)	66.2 (0.6)	50.4 (1.0)	57.6 (1.1)
$model_0$ mono.	15	41.6 (3.2)	64.1 (0.8)	61.3 (0.6)	59.0 (0.8)	43.1 (1.2)	56.7 (0.7)	61.4 (0.9)	45.7 (0.7)	57.6 (0.8)
$model_L$ mono.	1	41.8 (2.4)	65.5 (0.7)	63.7 (0.8)	61.6 (0.5)	44.2 (1.1)	57.6 (0.4)	64.6 (0.7)	49.5 (1.0)	56.0 (0.9)
	5	43.6 (1.8)	66.5 (0.5)	64.0 (0.6)	62.4 (0.6)	45.4 (0.7)	57.9 (0.5)	65.0 (0.8)	50.7 (0.7)	58.3 (0.9)

TABLEAU 5.8 – Mesures de F1-score sur les différents jeux de données d'évaluation $test_i$ pour les expériences sur l'étude des capacités des modèles continus sur MultiCoNER. Les informations supplémentaires données dans le tableau 5.7 s'appliquent ici aussi.

La comparaison entre $model_0$ multilingue et $model_L$ multilingue pour les deux corpus donne deux résultats intéressants. D'une part, nous observons qu'entraîner $model_L$ sur seulement une époque permet d'obtenir de meilleures performances que dans le cadre d'un entraînement monolingue ($model_0$ monolingue) et que ces performances sont proches des performances multilingues ($model_0$ multilingue) [10], alors que $model_0$ monolingue et multilingue sont tous les deux entraînés sur le nombre

10. À l'exception du chinois sur MultiCoNER, ce qui n'est pas surprenant étant donné que son transfert multilingue est négligeable.

maximum d'époques (50 ou 15). Cela signifie que model_L est capable d'atteindre de bonnes performances multilingues avec très peu d'entraînement, annulant ainsi l'effet de l'oubli. D'autre part, nous constatons que les performances multilingues du modèle continu model_L sont largement supérieures à celles de model_0 multilingue quand ils sont entraînés sur une seule époque d'apprentissage. Cela n'est pas surprenant étant donné que le classificateur est initialisé de manière aléatoire dans model_0. Néanmoins cela montre que le modèle est capable de retenir les connaissances des langues précédentes, bien qu'il ne soit pas clair si ces connaissances sont préservées dans le classificateur ou dans BERT.

Afin d'approfondir cette idée, nous entraînons model_L avec un classifieur aléatoire sur toutes les langues conjointement de la même manière que précédemment (voir « model_L + rnd clf multi. » dans les tableaux 5.7 et 5.8). Nous observons que les performances de ce modèle sont toujours largement supérieures aux performances de model_0 multilingue après une époque d'entraînement. Cependant pour MultiATIS++, ces performances ne sont pas aussi élevées que celles de model_L multilingue qui conserve son classificateur entraîné en continu. Pour MultiCoNER au contraire, ces performances sont presque identiques. Cela indique que la plupart des connaissances retenues des langues précédentes sont stockées dans BERT, et que les connaissances stockées dans le classificateur dépendent du corpus.

Dans l'ensemble, ces résultats nous amènent à penser que l'entraînement continu sur une séquence de langues permet en effet aux paramètres du modèle de tendre vers une meilleure initialisation multilingue.

5.7.2 Récupération des performances

Étant donné que l'entraînement continu permet de tendre vers une meilleure initialisation multilingue, nous explorons la possibilité de tirer parti de ce phénomène afin de récupérer rapidement les spécificités linguistiques perdues à cause du phénomène d'oubli (*fast recovery* [Hadsell et al., 2020]). Pour ce faire, nous entraînons par fine-tuning model_L sur la première langue de la séquence (entraînement monolingue « mono. ») une seconde fois (*i.* comme s'il s'agissait d'une $(L+1)^{\text{th}}$ langue) et évaluons le modèle sur la première langue uniquement. Comme précédemment model_L provient de nos expériences continues P_{1L}^{cont} mais les performances de ce modèle sont moyennées uniquement sur les 3 séquences commençant par la même langue. Les performances monolingues de model_L sont comparées aux performances monolingues de model_0, qui sont elles moyennées sur 5 graines aléatoires.

Comme le montrent les tableaux 5.7 et 5.8, en comparant model_L monolingue à model_0 monolingue (égale à la performance initiales sur la première langue P_{11}),

nous constatons que la performance de la première langue peut être récupérée et même améliorée avec un entraînement sur seulement une époque[3]. Ces résultats sont impressionnants sur MultiCoNER étant donné que l'oubli était important sur l'ensemble des langues (de -6,0 points de F1-score pour le néerlendais à -17,6 pour le chinois). Sur MultiATIS++, model_L monolingue atteint même les performances de model_0 multilingue dans la plupart des cas après une seule époque, ou bien présente une amélioration importante par rapport à model_0 monolingue. En particulier, l'hindi et le turc affichent des améliorations absolues de 3,9% et 7,8% respectivement par rapport au modèle monolingue model_0.

Notez que pour MultiATIS++, l'augmentation du nombre d'époques de récupération pour la première langue n'apporte pas d'améliorations considérables. La seule exception à cette observation est le turc, ce qui pourrait s'expliquer par la petite taille de son jeu de données d'apprentissage. Cependant, dans MultiCoNER, les performances s'améliorent encore après 5 époques, se rapprochant ainsi de la baseline multilingue. Étonnamment, model_L monolingue donne des performances équivalentes à la baseline multilingue pour le turc et le chinois.

Ces expériences ont donc montré que les modèles continus étaient capables de récupérer de manière efficace leurs performances initiales sur l'ensemble des langues. Cependant, bien que le coût de l'ajout d'une langue reste de $O(N)$, la possibilité de récupérer toutes les langues fait passer les coûts à $O(LN)$, ce qui rend son utilisation coûteuse en pratique. La conception d'une stratégie tirant pleinement parti de ces capacités de récupération pour limiter l'oubli à moindre coût est un sujet de recherche qu'il serait intéressant d'étudier.

5.8 Discussion

5.8.1 Synthèse des expériences

En résumé, nous observons pour les deux corpus un niveau élevé de transfert entre langues lors de l'apprentissage multilingue de la tâche d'étiquetage de séquences sur toutes les langues conjointement. Pour MultiATIS++ cependant, nous pensons que le transfert peut s'expliquer par le fait que les traductions conservent les mêmes valeurs de slots comme indiqué par [Do et al., 2020]. Cependant, cette adaptation des données à une autre langue n'est pas réaliste, puisqu'un système de dialogue est en pratique plus susceptible de gérer des entités spécifiques au pays dans lequel il est développé.

Cependant en pratique toutes les données de chaque langue ne sont pas forcément disponibles au moment où le modèle est entraîné pour la première fois. De plus,

adapter le modèle à une nouvelle langue dans le cadre d'un entraînement multilingue ou monolingue est coûteux en termes de ressources de calcul et de mémoire. Dans un contexte d'apprentissage continu où les langues sont apprises de manière séquentielle, ces coûts sont les plus faibles et le transfert entre langues est conservé sous la forme d'un transfert en avant. Cependant le modèle souffre du phénomène d'oubli sur la première langue de la séquence, particulièrement pour les langues à faibles ressources sur MultiATIS++ et sur l'ensemble des langues de MultiCoNER.

En examinant le transfert continu entre langues sur l'ensemble de la séquence, nous obtenons deux résultats surprenants. Premièrement, les mesures de transfert continu couramment utilisées peuvent ne pas être une estimation fiable de la distribution des performances entre les langues lorsque le transfert n'est pas distribué de manière égale. Puisque même dans d'autres axes d'adaptation, une variabilité considérable entre les jeux de données de données est à prévoir, nous pensons qu'une statistique comme la médiane pourrait être un meilleur choix, car nous pensons qu'elle représente mieux la performance attendue à un point donné. Deuxièmement, à mesure que la séquence progresse, le transfert vers l'avant s'améliore, tandis que le transfert vers l'arrière diminue. Cela pourrait indiquer que les paramètres du modèle restent une bonne initialisation pour les langues futures, mais que les spécificités des langues précédentes pourraient être perdues.

Motivés par cette hypothèse, nous comparons le modèle au début et à la fin de la séquence d'entraînement. Nos résultats suggèrent que la connaissance des langues passées est principalement stockée dans BERT multilingue (par opposition au classificateur spécifique à la tâche) et que le modèle peut effectivement tendre vers une meilleure initialisation multilingue, ce qui le rend apte à récupérer rapidement les performances perdues suite à l'oubli. Nous mesurons ensuite les capacités de récupération du modèle par rapport à la première langue de la séquence. Nous montrons empiriquement que la performance perdue peut être récupérée après un apprentissage sur seulement une époque, même si l'oubli est élevé. Les performances peuvent même être considérablement améliorées et se rapprocher de la baseline multilingue un apprentissage sur seulement une époque pour MultiATIS++ et 5 époques pour MultiCoNER.

À la lumière de ce qui précède, nous pensons que les méthodes d'apprentissage continu pourraient tirer profit des capacités de récupération de leur modèle (soit pour une seule langue, soit pour plusieurs langues conjointement) afin de limiter l'effet de l'oubli, tout en préservant ou même en augmentant le transfert vers l'avant.

5.8.2 Combiner apprentissage continu et apprentissage actif

Nous proposons ici une nouvelle direction de recherche à partir des résultats obtenus dans ce chapitre. Les expériences associées n'ont pas pu être réalisées par manque de temps mais il est intéressant d'en comprendre l'intérêt et de détailler la manière dont cela pourrait être mis en place.

Nous avons vu précédemment que l'apprentissage continu sur les langues des corpus MultiATIS++ et MultiCoNER permet un transfert en avant proche du transfert observé avec un entraînement multilingue, tout en minimisant les coûts d'entraînement dus à l'ajout d'une nouvelle langue. L'idée ici est de profiter de ce transfert, des langues déjà apprises vers de nouvelles langues, afin d'avoir recours à moins d'exemples d'entraînement lors de l'apprentissage de nouvelles langues. Nous faisons l'hypothèse ici que la diminution des exemples d'entraînement permettra de diminuer l'oubli catastrophique, présent à la fin de l'entraînement continu comme nous l'avons vu précédemment. Nous proposons ainsi d'ajouter à l'entraînement continu une composante d'apprentissage actif au sein de l'apprentissage d'une langue de la séquence. Pour chaque langue de la séquence autre que la première, lors de l'apprentissage de chaque langue i, considérons que l'on ne dispose que d'un petit jeu de données de $train_i$ et de dev_i annotées A_i et d'un grand jeu de données de $train_i$ et de dev_i non-annotées U_i. La combinaison de l'entraînement continu et de l'apprentissage actif pourrait se dérouler selon les étapes suivantes :

1. *Initialisation.* Le modèle est entraîné sur le jeu de données d'entraînement d'une langue source choisie, résultant en un modèle m_i (avec i=0 ici).
2. *Apprentissage de la langue i (i = i + 1).* On considère maintenant la langue suivante dans la séquence. Le modèle m_{i-1} est adapté sur A_i, résultant en un modèle $m_{i,j}$ (avec j=0 ici).
3. *Boucle d'apprentissage actif (j = j +1).* On sélectionne selon une *stratégie de sélection* choisie, k exemples de U_i à annoter qu'on rajoute à A_i (et qu'on enlève à U_i).
4. On adapte le modèle m_{i-1} sur A_i, résultant en un modèle $m_{i,j}$.
5. On répète les deux étapes précédentes tant qu'il reste des exemples dans U_i et que le *critère d'arrêt* n'est pas vérifié.
6. On répète toutes les étapes à partir de l'étape 2 jusqu'à ce qu'on arrive à la dernière langue de la séquence.

Au cours des expériences, pourront être testées plusieurs stratégies de sélection et plusieurs critères d'arrêt différents. La stratégie de sélection pourra s'appuyer sur une mesure d'incertitude des prédictions du modèle $m_{i,j}$ sur U_i, en prenant les k premiers énoncés utilisateur de U_i pour lesquels la moyenne de l'entropie

de la distribution de probabilité prédite par le modèle $m_{i,j}$ sur l'ensemble des tokens de l'énoncé est maximale. Pour le critère d'arrêt, il pourra être vérifié si la différence de performance entre $m_{i,j}$ et $m_{i,j-1}$ sur le jeu de données de dev de A_i est inférieure à un seuil t. Le processus décrit précédemment pourra aussi être modifié, en considérant le scénario où des exemples d'entraînement non-annotés de chaque langue arrivent mélangés au cours du temps.

Le fait de combiner l'apprentissage continu et l'apprentissage actif, est d'autant plus intéressant qu'il a l'avantage de rendre le système autonome, puisque c'est lui qui décidera des nouveaux exemples d'entraînement à annoter ainsi que de l'arrêt de l'entraînement pour chaque langue. Cette idée se rapproche de fait du LL. De plus cette approche n'a été étudiée, à notre connaissance, que dans des contextes précis, notamment dans le domaine robotique [Qureshi et al., 2020] et le domaine médical [Zhou et al., 2021].

5.9 Conclusion

Dans ce chapitre, nous avons présenté une analyse du transfert entre langues dans le cadre de l'apprentissage continu pour la tâche d'étiquetage de séquences en utilisant le modèle multilingue BERT [Devlin et al., 2019b] ainsi que les corpus MultiATIS++ [Xu et al., 2020b] et MultiCoNER [Malmasi et al., 2022a]. La tâche d'étiquetage de séquences est la tâche générale à réaliser pour la détection des slots qui nous intéresse particulièrement dans le cadre de ce mémoire pour les systèmes de dialogue orientés tâche. En plus d'être directement liée à l'étude du LL, cette manière d'étudier le transfert entre langues est à la fois novatrice et adaptée à un cas réel, ce qui explique son intérêt.

Notre principale conclusion suggère que, bien que l'oubli soit présent, le transfert entre langues est conservé sous la forme d'un transfert vers l'avant, ce qui permet au modèle d'avoir des capacités de récupération substantielles. De plus, nous montrons empiriquement que 1) un transfert vers l'avant élevé est lié au fait que les paramètres du modèle tendent progressivement vers une meilleure initialisation multilingue, et 2) que la plupart des connaissances des langues passées sont stockées dans l'encodeur de représentation des tokens (BERT) et non dans le classificateur spécifique à la tâche. Enfin, nous constatons également que les métriques actuelles pour évaluer le transfert dans le cadre de l'apprentissage continu doivent être adaptées si nous voulons mieux estimer la distribution du transfert sur l'axe d'adaptation.

Ces conclusions et ces observations sont intéressantes pour le LL : elles montrent l'importance de diversifier les cadres d'études, mais aussi le fait que des travaux

sont encore nécessaires pour analyser et comprendre comment le transfert en avant et le transfert en arrière se comportent et comment contrer le phénomène d'oubli de manière efficace. Elles mettent aussi en lumière de nouvelles directions de recherche liées au LL, comme la combinaison de l'apprentissage continu et de l'apprentissage actif.

Chapitre 6

Étude du transfert entre domaines dans le cadre de l'apprentissage zero-shot

6.1 Introduction

Dans les chapitres précédents, nous nous sommes d'abord intéressés à l'apprentissage sur le terrain de nouvelles instances, puis à l'étude du transfert entre langues dans le cadre de l'apprentissage continu sur une séquence de langues. Ces deux chapitres se concentraient sur la tâche de détection des slots, tâche qui ne nécessite pas forcément de prendre en compte le contexte du dialogue. Ainsi, nous nous intéressons ici à une tâche plus complexe qui nécessite précisément de considérer l'ensemble du dialogue. Cette tâche correspond à la tâche de suivi de l'état du dialogue (DST) comme décrit en section 2.2.2. Son objectif est de permettre au système de dialogue d'avoir accès à une représentation du but courant de l'utilisateur, notamment via les critères que l'utilisateur a mentionné ou confirmé. Dans sa forme la plus étudiée, cette tâche consiste, à chaque tour du dialogue, à prédire la valeur de chacun des types de slots connus par le système afin de représenter les critères courants de l'utilisateur [Jacqmin, 2022]. L'état du dialogue correspond alors à un ensemble de paires `(type de slot, valeur de slot)`. Cette tâche est illustrée à l'aide d'un exemple en Figure 6.1.

Le corpus le plus utilisé pour étudier la tâche de DST est le corpus MultiWOZ [Budzianowski et al., 2018a]. En effet, ce corpus est intéressant car il contient un nombre important de dialogues sur des domaines différents. La présence de plusieurs domaines, dont les types de slots peuvent être différents d'un domaine

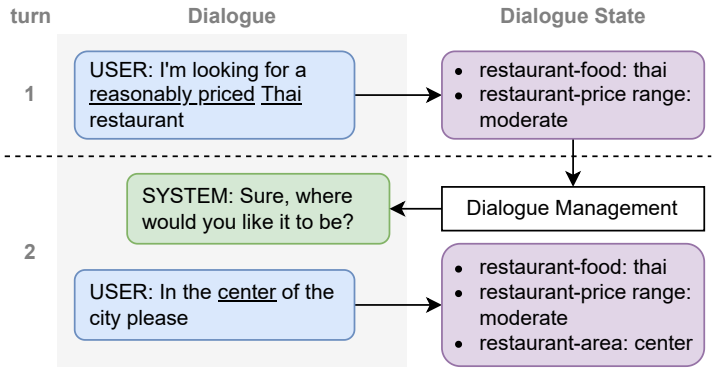


FIGURE 6.1 – Exemple d'un dialogue entre un utilisateur et un système de dialogue orienté tâche dans le domaine des restaurants, et état du dialogue associé à chaque tour du dialogue.

à l'autre, permet notamment l'étude du transfert entre domaines dans le cadre de différentes configurations expérimentales. Après avoir étudié le transfert entre langues dans le cadre de l'apprentissage continu, nous nous concentrons ainsi dans ce chapitre sur l'étude du transfert entre domaines dans le cadre de l'apprentissage zero-shot. Ce cadre expérimental est aussi intéressant du point de vue du LL puisqu'il s'intéresse à la capacité d'un système à s'adapter, sans entraînement supplémentaire, à de nouveaux éléments jamais vus pendant l'entraînement initial, ceci en profitant du transfert entre domaines. Nous réalisons cette étude sur un modèle de DST existant nommé SUMBT [Lee et al., 2019]. Ce modèle repose sur BERT [Devlin et al., 2019b], un modèle de langue pré-entraîné de manière auto-supervisée sur un large corpus de textes, correspondant à la partie encodeur de l'architecture `transformer` comme vu en section 2.1.3. Dans un premier temps, nous nous intéressons aux capacités de généralisation de SUMBT sur les valeurs de slots inconnues et étudions le transfert entre types de slots partageant les mêmes valeurs de slots. Ensuite, nous étudions le transfert zero-shot entre domaines dans le cadre de la configuration « leave-one-out », où le modèle est entraîné sur toutes les données MultiWOZ sauf celles associées au domaine cible, puis évalué sur les données de test du domaine cible. Dans cette configuration, le modèle doit être en mesure de prédire des types de slots qu'il n'a jamais vu pendant l'entraînement. Cette capacité n'est possédée que par une minorité des modèles de DST développés aujourd'hui, minorité dont fait partie SUMBT, d'où notre intérêt pour ce modèle. Dans le cadre de ces expériences nous cherchons à modifier l'architecture SUMBT et à mettre au point des méthodes afin d'améliorer ses performances zero-shot.

Dans un premier temps, nous réalisons en section 6.2 un état de l'art des travaux s'intéressant à l'étude les capacités de généralisation et de transfert de modèles de

DST dans diverses configurations, afin de positionner nos expériences par rapport aux travaux existants. Ensuite, en section 6.3 nous présentons le corpus MultiWOZ en détails ainsi que quelques travaux réalisés sur la correction de ses annotations. En section 6.4 nous présentons l'architecture du modèle SUMBT ainsi que la manière dont le modèle est entraîné et son fonctionnement lors de l'inférence. Par la suite, nous présentons en section 6.5 notre étude des capacités de généralisation et de transfert sur des valeurs de slots inconnues de SUMBT. Enfin, en section 6.6 nous proposons des variantes de l'architecture SUMBT ainsi qu'une méthode appelée *modulation de l'attention*, à appliquer pendant l'évaluation afin d'améliorer les performances zero-shot de SUMBT dans le cadre du transfert sur de nouveaux domaines.

6.2 Etat de l'art

6.2.1 Étude des capacités de généralisation

Lors de l'application d'un modèle à la tâche de DST du corpus MultiWOZ, certains travaux se sont intéressés aux capacités de généralisation de leur modèle sur des valeurs de slots jamais vues pendant l'entraînement. Ainsi, le modèle TripPy [Heck et al., 2020], présenté de manière succincte en section 2.2.2, semble montrer de bonnes capacités de généralisation lorsqu'il est évalué sur le jeu de données *test* auquel 84% des paires uniques ont été remplacées par des nouvelles valeurs. Cependant ce n'est pas le cas de tous les modèles. En effet, [Qian et al., 2021] ont évalué le modèle DST-BART [Lewis et al., 2020] sur le jeu de données *test* de MultiWOZ 2.1, auquel l'ensemble des valeurs de slot a été remplacé par des valeurs inconnues et ils ont constaté une perte de JGA de 29 points par rapport au jeu de données *test* original. Ceci montre que le modèle a tendance à sur-mémoriser les valeurs qui apparaissent dans le jeu de données d'entraînement, ce qui résulte en de faible capacités de généralisation. Les auteurs supposent que cet effet est renforcé à cause des biais présents dans le corpus MultiWOZ puisqu'ils observent par exemple que la valeur `cambridge` est associée dans 50% des cas au type de slot `destination` dans le domaine `train`.

Par ailleurs, lors de l'analyse de leurs résultats, plusieurs travaux à l'image de [Lee et al., 2019], supposent que leur modèle profite du transfert entre types de slots à travers le partage des connaissances associées à chaque type de slot. Cependant, à notre connaissance aucune étude ne s'est intéressée à montrer que ce transfert existe réellement. Ainsi, en section 6.5 nous nous intéressons aux capacités de généralisation sur des valeurs de slots inconnues et aux capacités de transfert entre types de slots partageant des valeurs communes, ces capacités étant évaluées sur le modèle SUMBT.

6.2.2 Étude des capacités de transfert zero-shot entre domaines

Au fur et à mesure des années, de nombreux modèles ont été proposés et appliqués à la tâche de DST de MultiWOZ. Cependant, seulement une minorité de ces modèles sont utilisables dans le cas où le modèle doit détecter des slots qu'il n'a jamais vu pendant l'entraînement. Récemment, de nombreux travaux se sont intéressés au transfert vers un domaine cible inconnu dans le cadre de l'apprentissage zero-shot. Dans le cas du corpus MultiWOZ, on parle aussi de configuration « leave-one-out », puisque le modèle est entraîné sur les dialogues de MultiWOZ qui ne concernent pas le domaine cible, puis est évalué sur les dialogues du domaine cible. Le corpus Schema-Guided Dialogue [Rastogi et al., 2020] est également utilisé pour évaluer les capacités de transfert, cependant ce corpus ne distingue que slots inconnus des slots vu pendant l'entraînement, et ne permet donc pas l'analyse du transfert entre-domaines.

[Wu et al., 2019] sont les premiers à s'intéresser à la configuration « leave-one-out » sur MultiWOZ avec leur modèle TRADE (Transferable Dialogue State Generator). Leur modèle repose sur un encodeur d'énoncés construit sur un RNN, un portail pour les types de slots (*slot gate*) et un générateur d'état partagé entre tous les domaines. Par la suite, le modèle SUMBT [Lee et al., 2019] est proposé puis appliqué aux expériences zero-shot [Campagna et al., 2020]. Comme vu précédemment, ce modèle repose sur le modèle de langue pré-entraîné BERT [Devlin et al., 2019a] et un RNN. Cependant, au lieu de construire de nouvelles architectures, les modèles les plus récents, qui présentent des performances à l'état de l'art, exploitent directement des modèles de langue génératifs pré-entraînés sur un large corpus de textes, comme GPT-2 [Radford et al., 2019] ou T5 [Raffel et al., 2020]. Ils travaillent alors sur la forme de l'entrée elle-même en incorporant des descriptions de slot [Lin et al., 2021b, Zhao et al., 2022], en montrant des exemples annotées [Gupta et al., 2022], ou en considérant un type de slot comme une question [Li et al., 2021, Lin et al., 2021a].

Ainsi dans ce chapitre nous cherchons à voir si les performances des premiers modèles développés pour le transfert zero-shot entre domaines peuvent être améliorées, notamment en modifiant leur architecture. Nous nous intéressons en particulier au modèle SUMBT afin de faire suite aux premières expérimentations sur ce modèle visant à étudier ses capacités de généralisation sur de nouvelles valeurs de slot.

6.3 MultiWOZ

Multi-Domain Wizard-of-Oz dataset (MultiWOZ) [1] est un corpus regroupant des conversations écrites entre un système de dialogue orienté tâche et un utilisateur. Les conversations ont été collectées dans le cadre d'une configuration dite « du magicien d'Oz », où les rôles de l'utilisateur et du système sont tous les deux joués par des humains. Pour chaque dialogue, une description détaillée en langage naturel du but de l'utilisateur était fournie à la personne jouant le rôle de l'utilisateur et la personne jouant le rôle du système avait accès à la base de donnée associée. La première version du corpus multi-domaine [Budzianowski et al., 2018b] contient plus de 10 000 dialogues couvrant sept domaines différents : `attraction`, `bus`, `hospital`, `hotel`, `restaurant`, `taxi` et `train`. Parmi ces dialogues, 3 406 concernent un unique domaine, tandis que 7 032 concernent au moins deux domaines différents. Les dialogues ont été séparés aléatoirement en trois jeux de données *train* (8 438 dialogues), *dev* (1 000 dialogues) et *test* (1 000 dialogues).

Dans le cadre des expériences décrites dans ce chapitre, nous n'utilisons pas les données des domaines `bus` et `hospital` puisqu'aucune donnée d'évaluation n'existe pour ces domaines. Pour les domaines restants nous récapitulons dans le tableau 6.1 les différents types de slots associés.

Après avoir constaté de nombreuses erreurs dans les annotations réalisées dans MultiWOZ 2.0 au niveau des état du dialogue et des énoncés, le corpus été corrigé et en deux versions consécutives MultiWOZ 2.1 [Eric et al., 2020] et MultiWOZ 2.2 [Zang et al., 2020]. Dans ce chapitre, nous utilisons la version 2.0 de MultiWOZ puisque le modèle SUMBT [Lee et al., 2019], sur lequel se porte nos expériences, a originellement été entraîné sur cette version. Cependant, dans la section 6.6 nous réalisons aussi les expériences sur la version 2.1 afin de comparer nos résultats à ceux d'autres travaux.

6.4 SUMBT

Nous décrivons ici de manière synthétique l'architecture du modèle « Slot-Utterance Matching Belief Tracker » (SUMBT) [Lee et al., 2019] que nous utilisons dans le carde de nos expériences, ainsi que la façon dont il est entraîné et comment il fonctionne pendant l'inférence. Pour plus d'informations, nous vous encourageons à lire l'article original.

1. Voir https://github.com/budzianowski/multiwoz.

Domaine	Types de slots #	Liste
attraction	3	area, name, type
hotel	10	area, book day, book people, book stay, internet, name, parking, price range, stars, type
restaurant	7	area, book day, book people, book time, food, name, price range
taxi	4	arrive by, leave at, departure, destination
train	6	arrive by, book people, day, departure, destination, leave at

TABLEAU 6.1 – Types de slots par domaine pour les domaines de MultiWOZ 2.0 disposant de données d'évaluation.

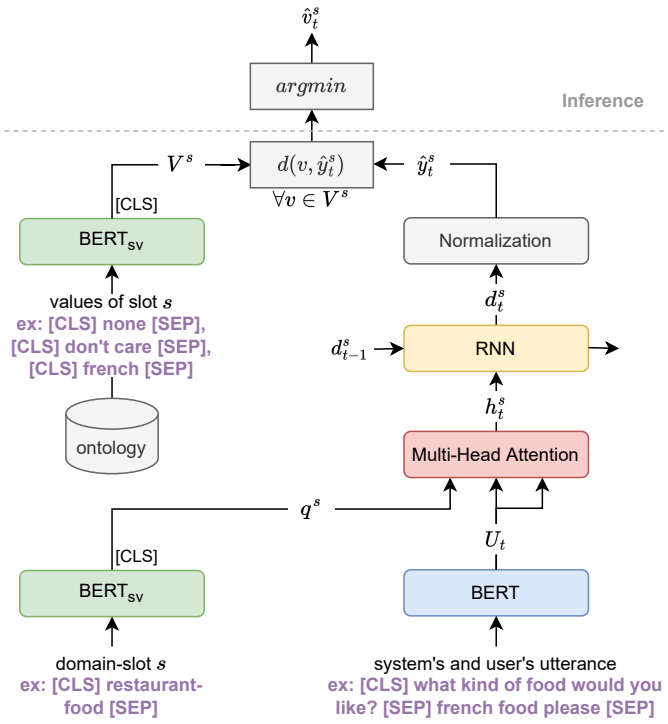


FIGURE 6.2 – Architecture du modèle SUMBT [Lee et al., 2019]

6.4.1 Architecture

L'idée principale de SUMBT est de faire correspondre à chaque type de slot une valeur de slot à partir d'une ontologie et d'un tour de dialogue (énoncé d'un système et d'un utilisateur). L'architecture du modèle est illustrée dans la figure 6.2.

Le texte correspondant à la concaténation du nom du domaine et du nom du

type de slot (`[CLS] <domaine>-<nom du type de slot> [SEP]`, noté s dans la figure 6.2) est d'abord encodé par un modèle noté $BERT_{sv}$, qui est initialisé avec les poids du modèle pré-entraîné BERT [Devlin et al., 2019a]. La représentation du token `[CLS]` issue de $BERT_{sv}$ est ensuite récupérée afin d'obtenir une représentation globale du type de slot en entrée sous forme d'un vecteur q^s. D'une manière similaire, le texte correspondant à l'énoncé du système et de l'utilisateur est encodé par un modèle noté $BERT$, initialisé lui aussi avec les poids de BERT [Devlin et al., 2019a]. Ainsi, chaque token du tour de dialogue en entrée est représenté par un vecteur contextuel résultant en la matrice U_t. Notez que $BERT$ est adapté durant l'entraînement tandis que les poids de $BERT_{sv}$ sont fixés pendant l'entraînement.

Par la suite, la représentation du slot q^s est utilisée comme *requête* par la couche d'attention multi-têtes (voir section 2.1.2) tandis que la représentation du tour de parole U_t est utilisée à la fois comme *clé* et comme *valeur*. Cette couche d'attention multi-tête permet au modèle d'attirer son attention sur les tokens qui sont liés au slot passé en requête et de produire une représentation globale des tokens sur lesquels s'est porté son attention.

Puisque la tâche de DST consiste à mettre à jour l'état du dialogue courant, le modèle a besoin d'avoir accès aux informations sur l'état du dialogue précèdent. L'accès à ces informations se fait grâce à un RNN qui prend en entrée la sortie de la couche d'attention multi-tête h_t^s, ainsi que la sortie du RNN du tour de dialogue précédent d_{t-1}^s. La sortie du RNN est ensuite normalisée pour produire $\hat{y}_t^s$, la prédiction de la représentation de la valeur associée au type de slot s en entrée du modèle.

En parallèle, l'ensemble des valeurs du type de slot s est encodé par $BERT_{sv}$ en récupérant uniquement la représentation du token `[CLS]` pour chaque valeur, résultant en une matrice V^s. La distance euclidienne entre la prédiction $\hat{y}_t^s$ et chaque représentation $v \in V^s$ est ensuite calculée. Comme décrit dans la partie suivante, la manière dont cette distance est ensuite utilisée dépend de si le modèle est entraîné ou s'il est évalué.

6.4.2 Entraînement et inférence

Au cours de l'apprentissage, le modèle apprend à minimiser la distance entre $\hat{y}_t^s$ et y_t^s et à maximiser la distance entre $\hat{y}_t^s$ et les valeurs $v \neq y_t^s$, y_t^s correspondant à la valeur de référence associée au type de slot s en entrée du modèle. Cet apprentissage se fait via la minimisation de l'entropie croisée.

L'entraînement se fait sur 300 époques, sachant qu'il est configuré de manière à s'arrêter quand plus aucune amélioration n'est visible sur le jeu de données de

validation (*early stopping*). Le meilleur modèle est sélectionné sur ce même jeu de données de validation. L'optimisation se fait grâce à l'optimiseur Adam avec *weight decay* [Loshchilov and Hutter, 2019] et un taux d'apprentissage configuré sur la valeur 5×10^{-5} avec un planificateur linéaire avec *warmup*. Notez que dans le cadre des expériences décrites dans ce chapitre, nous utilisons les mêmes hyper-paramètres et les mêmes conditions d'entraînement.

Lors de l'inférence, la valeur $v \in V^s$ dont la distance est la plus faible avec la sortie du modèle $\hat{y}_t^s$, est donnée comme prédiction de la valeur associée au type de slot s et au tour de dialogue considérés en entrée. De cette manière, lors de l'inférence, de nouvelles valeurs de slot peuvent être prédites et n'importe quel type de slot peut être utilisé en entrée comme *requête*, tant que l'ontologie contient la liste des valeurs associées à chaque type de slot. Les modèles SUMBT entraînés peuvent donc être appliqués à la fois à de nouvelles valeurs et à de nouveaux domaines après la mise à jour de l'ontologie, et les modèles peuvent ainsi prédire de nouvelles valeurs et de nouveaux slots jamais vus pendant l'entraînement.

6.5 Prédire de nouvelles valeurs de slot

Dans cette section nous étudions les capacités de généralisation et de transfert du modèle SUMBT pour la détection de nouvelles valeurs de slot. Plus précisément, nous nous posons les questions suivantes :

Q 1 *À quel point SUMBT est-il en mesure de généraliser sur les paires* `(type de slot, valeur de slot)` *inconnues, sachant que le type de slot a été rencontré pendant l'entraînement mais avec des valeurs de slots différentes ?*

Q 2 *À quel point SUMBT permet-il le transfert entre slots et, plus précisément, à quel point la rencontre pendant l'entraînement de la valeur v_k avec un type de slot t_i, aide-t-elle le modèle à associer durant l'évaluation la même valeur v_k à un autre type de slot $t_{i \neq j}$, sachant que la paire $((t_{i \neq j}, v_k))$ n'a jamais été vue pendant l'entraînement ?*

Pour ce faire nous présentons en section 6.5.1 la configuration des expériences incluant la préparation des données, les données sur lesquelles chaque modèle est entraîné ainsi que la manière avec laquelle sont évaluées les capacités de généralisation et de transfert. Ensuite, en section 6.5.3 nous présentons les résultats de notre étude sur les capacités de généralisation et de transfert de SUMBT. Enfin, en section 6.5.4, nous discutons les résultats obtenus vis-à-vis du LL.

6.5.1 Configuration des expériences

Pour étudier les deux questions de recherche énoncées précédemment, il est nécessaire d'adapter les données MultiWOZ. Pour ce faire, plusieurs approches sont possibles. De nouvelles valeurs de slots qui n'apparaissent pas dans les données MultiWOZ peuvent être collectées afin de remplacer les valeurs de slots dans les données de test par des valeurs inconnues [Qian et al., 2021]. Cette approche permet de comparer les performances d'un modèle sur deux jeux de données de test dont les dialogues sont les mêmes, mis à part au niveau des valeurs de slots. Ceci permet d'isoler l'impact des valeurs de slots inconnues, cependant la collecte des valeurs peut être coûteuse et les dialogues résultant du remplacement des valeurs peuvent ne pas être réalistes. Une autre approche consiste à sélectionner des paires de slots directement à partir de l'ontologie du corpus et les définir comme étant inconnues. Ainsi, les jeux de données originaux *train*, *dev* et *test* de MultiWOZ sont divisés entre les dialogues contenant uniquement des paires de slot définies comme connues et les dialogues contenant au moins une paire de slot définie comme inconnue. Par la suite, les capacités de généralisation sont évaluées notamment en entraînant le modèle uniquement sur les paires de slots définies comme connues, puis en comparant ses performances sur les paires connues à celles sur les paires inconnues. Cette approche, contrairement à la première, permet d'évaluer le transfert. En effet, les paires de slots de l'ontologie définies comme connues peuvent elles aussi être divisées de manière à séparer les paires pour le transfert, dont la valeur apparaît dans l'ensemble des paires inconnues, des autres paires. Par la suite, les capacités de transfert sont évaluées en entraînant un modèle sur les paires connues, incluant les paires pour le transfert, et un autre modèle sur les paires connues, excluant les paires pour le transfert, puis en évaluant ces deux modèles sur les paires inconnues et en comparant les performances.

Préparation des données

L'ontologie et les ensembles de données originaux *train*, *dev* et *test* de MultiWOZ au format demandé par SUMBT [2] sont d'abord divisés en trois sous-ensembles que nous distinguons avec les mentions *known*, $transfert$ et *unknown*. Pour plus de lisibilité les mention *known*, $transfert$ et *unknown* sont abrégées respectivement K, T et U et les jeu de données originaux portent la mention O.

L'ontologie originale $ontology_O$ est ainsi séparées en trois ontologies $ontology_K$, $ontology_T$ et $ontology_U$ selon les propriétés suivantes :

Propriété 1 *Les ontologies sont toutes disjointes en terme de paire* `(type, valeur)`*, telles que :*

2. Voir https://github.com/SKTBrain/SUMBT.

— $ontology_K \cap ontology_T = \emptyset$
— $ontology_T \cap ontology_U = \emptyset$
— $ontology_K \cap ontology_U = \emptyset$

Propriété 2 *Aucune des valeurs des ontologies unknown et $transfert$ n'est présente dans l'ontologie known, telles que :*
— $\{v|(t,v) \in ontology_K\} \cap \{v|(t,v) \in ontology_T\} = \emptyset$
— $\{v|(t,v) \in ontology_K\} \cap \{v|(t,v) \in ontology_U\} = \emptyset$

Propriété 3 *Les valeurs de l'ontologie $transfert$ sont comprises dans l'ontologie unknown, telles que :*
$\{v|(t,v) \in ontology_T\} \subset \{v|(t,v) \in ontology_U\}$

Les paires de l'ontologie *unknown* sont d'abord choisies aléatoirement dans l'ontologie originale. Les ontologies $transfert$ et *known* sont ensuite construites de manière à respecter les propriétés énoncées précédemment.

Pour séparer les jeux de données originaux *train*, *dev* et *test* en leurs sous-ensembles *unknown*, $transfert$ et *known*, nous parcourons l'ensemble des dialogues de chaque jeu de données et procédons de la manière suivante : si au moins un tour du dialogue contient une paire de l'ontologie *unknown*, le dialogue est transféré dans les données *unknown* ; sinon si au moins un tour du dialogue contient une paire de l'ontologie $transfert$, le dialogue est transféré dans les données $transfert$; sinon le dialogue est transféré dans les données *known*.

Modes d'entraînement et modèles

Nous décrivons ici les différentes manières d'entraîner le modèle SUMBT [Lee et al., 2019] dans le cadre de nos expériences ainsi que les modèles en résultant. Les conditions d'entraînement sont les mêmes que celles décrites en section 6.4.2, en dehors du fait que les différents modèles sont entraînés et évalués sur des jeux de données et des ontologies différentes. Afin de réaliser nos expériences, nous entraînons 3 modèles principaux :
— $model_I$: un modèle dit « ignorant », entraîné sur les jeux de données $train_K$ et dev_K à l'aide de l'ontologie $ontology_K$;
— $model_{PK}$: un modèle dit « partially-knowing », entraîné sur les jeux de données $train_{K+T}$ et dev_{K+T} à l'aide de l'ontologie $ontology_{K+T}$; et,
— $model_{AK}$: un modèle dit « all-knowing », entraîné sur les jeux de données $train_{K+T+U}$ et dev_{K+T+U} à l'aide de l'ontologie $ontology_{K+T+U}$

Pour évaluer si l'ontologie utilisée par le modèle pendant l'entraînement a une influence sur les performances du modèle, nous entraînons aussi 3 autres modèles :
— $model_{2I}$: un modèle dit « twice ignorant », entraîné sur les jeux de données $train_K$ et dev_K à l'aide de l'ontologie issue des données $train_K$ et dev_K ;

	data			**ontology**				
	K	$K+T$	$K+T+U$	$train_K$ $+dev_K$	$train_K$ $+dev_K$ $+test_K$	K	$K+T$	$K+T+U$
$model_{2I}$	×			×				
$model_{EI}$	×				×			
$model_I$	×					×		
$model_{SI}$	×							×
$model_{PK}$		×					×	
$model_{AK}$			×					×

TABLEAU 6.2 – Données et ontologies utilisées pour entraîner chaque modèle pour l'étude des capacités de généralisation et de transfert de SUMBT [Lee et al., 2019] sur des nouvelles valeurs de slot.

— $model_{EI}$: un modèle dit « enough ignorant », entraîné sur les jeux de données $train_K$ et dev_K à l'aide de l'ontologie issue des données $train_K$, dev_K et $test_K$; et,
— $model_{SI}$: un modèle dit « semi ignorant », entraîné sur les jeux de données $train_K$ et dev_K à l'aide de l'ontologie $ontology_{K+T+U}$.

L'ensemble des modèles ainsi que les données utilisées pour leur entraînement est résumé en table 6.2. Ces modèles sont entraînés sur onze graines aléatoires différentes.

Évaluation des capacités de généralisation et de transfert

Pour l'évaluation, nous utilisons la Joint Goal Accuracy (JGA) qui est la métrique de référence pour la tâche de DST. Celle-ci indique le pourcentage de tours de dialogue qui ont été prédits correctement, c'est-à-dire dans notre cas, les tours pour lesquels toutes les paires (types, value) ont été prédites correctement (paires, où la valeur de référence est `none`, incluses). Nous indiquons aussi parfois la *slot accuracy*, qui correspond au pourcentage de valeurs correctement prédites sur l'ensemble des tours pour un type de slot donné [3].

Nous rappelons que l'ontologie sur laquelle repose le modèle peut être mise à jour lors de l'évaluation afin de prédire des valeurs de slot jamais vues pendant l'entraînement. Plus précisément, celle-ci est mise à jour en fonction du jeu de données de test sur lequel se fait l'évaluation. Par exemple, si nous évaluons $model_I$ - qui a été entraîné avec $ontologie_K$ - sur $test_{K+T}$, il faudra d'abord mettre à jour l'ontologie avec les paires de $ontology_T$.

3. La valeur `none` a donc autant de poids qu'une autre valeur.

	train	*dev*	*test*
K	2 546	164	176
T	2 579	369	360
U	3 313	467	464
$K+T$	5 125	533	536
$T+U$	5 892	836	824
$K+T+U$	8 438	1 000	1 000

TABLEAU 6.3 – Nombre de dialogues dans chaque jeu de données généré à l'aide de notre méthode pour une proportion de valeurs connues pour chaque type de slot de 0,14 et une graine aléatoire de 1. Nous pouvons par exemple lire que $train_K$ est composé de 2 546 dialogues.

Nous évaluons d'abord les trois modèles principaux $model_I$, $model_{PK}$ et $model_{AK}$ sur $test_K$, afin de voir si le nombre de données d'entraînement a un impact sur les performances des modèles. Ensuite, pour répondre à la question Q1, nous évaluons $model_I$ et $model_{AK}$ sur $test_{T+U}$ et nous comparons les performances de $model_I$ sur $test_K$ et $test_{T+U}$ ainsi que les performances de $model_I$ et $model_{AK}$ sur $test_{T+U}$. Enfin, pour répondre à la question Q2, nous évaluons les 3 modèles principaux sur $test_U$ et observons l'écart de performances entre ces trois modèles.

Enfin, pour évaluer l'impact de l'ontologie pendant l'entraînement, nous comparons les performances de modèles $model_{2I}$, $model_{EI}$, $model_I$ et $model_{SI}$ sur $test_{T+U}$. Étant donné que nous optimisons l'entraînement via la minimisation de l'entropie croisée, le modèle est supposé apprendre à mieux discriminer les représentations des différentes slot-values au fur et à mesure de son entraînement. Nous supposons donc que parmi deux modèles entraînés sur les mêmes données, celui qui aura eu connaissance pendant l'entraînement d'une ontologie plus fournie que l'autre obtiendra des meilleures performances sur des données contenant des paires qui ne faisaient pas partie des données d'entraînement, c'est-à-dire que $P_{model_{2I}}(test_{T+U}) < P_{model_{EI}}(test_{T+U}) < P_{model_I}(test_{T+U}) < P_{model_{SI}}(test_{T+U})$.

6.5.2 Analyse d'un corpus de données généré

Nous avons séparé les jeux de données originaux *train*, *dev* et *test* en leurs sous-ensembles *unknown*, $transfert$ et *known* comme décrit en section 6.5.1 pour une proportion de valeur connue pour chaque slot de 0,14 et une graine aléatoire de 1. Nous analysons dans cette section les jeux de données obtenus.

Étant donné les données originales et la méthode choisie pour générer les données, des paires de l'ontologie *known* peuvent ne pas être présentes dans les données

d'entraînement *known*. En effet, il se peut que l'ensemble des dialogues contenant une paire de l'ontologie *known* soit dans les données $transfert$ ou *unknown* parce que ces dialogues contiennent au moins une paires des ontologies $transfert$ ou *unknown*. Parmi les paires de $ontology_K$, 42% ne sont pas dans $train_K$, 39% ne sont pas dans les jeux de données $train_K$ et dev_K et 37% ne sont pas dans les jeux de données $train_K$, dev_K et $test_K$. Ces proportions sont importantes mais possibles dans un cas réel où l'ontologie utilisée pour l'entraînement et l'évaluation aurait été récupérée à partir des données d'entraînement et de test mais aussi à partir de ressources extérieures (*e.g.* base de données, Wikipedia). Nous observons notamment que 3,6% des tours de $test_K$ comportent au moins une paire qui n'est pas dans l'ensemble des données $train_K$ et dev_K, contre 0,3% pour les données $train_{K+T+U}$ et dev_{K+T+U}. Ceci signifie que nous pouvons nous attendre à une perte de performance de $model_I$ de maximum 3,3% sur $test_K$ par rapport à $model_{AK}$, si le modèle n'est pas en mesure de généraliser sur des paires qu'il n'a jamais vu dans ses données d'entraînement et qu'avoir connaissance de ces paires pendant l'entraînement à travers l'ontologie ne suffit pas pour généraliser.

La perte de performance pourrait être encore plus élevée étant donné que le nombre de données d'entraînement pour $model_I$ est plus faible que pour les deux autres modèles (voir tableau 6.3). Nous observons plus précisément que 25,35% des 44 146 tours de $train_{T+U}$ contiennent uniquement des paires inclues dans les données $test_K$ et que 1,55% contiennent uniquement les valeurs `none` et `don't care`. Étant donné que $model_I$ aura eu moins de données d'entraînement contenant des paires apparaissant dans les données $test_K$ que $model_{PK}$ et $model_{AK}$ nous pouvons ainsi nous attendre à un écart de performance.

Pour ce qui est du transfert, nous observons qu'environ 5,97% des tours de $test_U$ contiennent des valeurs qui ne sont pas présentes dans les données $train_{K+T}$ et dev_{K+T}, c'est-à-dire que le transfert n'est pas possible sur ces tours et que si le modèle n'est pas en mesure de généraliser sur les paires dont les valeurs ne sont pas dans les données d'entraînement, nous pouvons nous attendre à une différence de performance entre $model_{PK}$ et $model_{AK}$ de maximum 5,97%.

6.5.3 Résultats

Les performances des différents modèles décrits en section 6.5.1 sont présentées dans les tableaux 6.4.

Écarts de performance sur les paires connues de tous

Nous nous concentrons d'abord sur les performances des trois modèles principaux sur $test_K$, contenant des dialogues associés à des paires de slots vues pendant l'en-

		$test_K$	$test_{K+T}$	$test_{T+U}$	$test_U$	$test_{K+T+U}$
$model_{2I}$	JGA	41.43 (±2.27)	25.70 (±1.12)	17.81 (±0.62)	16.85 (±0.69)	21.30 (±0.86)
	Acc.	96.40 (±0.28)	93.43 (±0.23)	90.94 (±0.23)	90.05 (±0.27)	91.76 (±0.23)
$model_{EI}$	JGA	42.24 (±2.09)	25.92 (±1.06)	17.94 (±0.62)	17.11 (±0.62)	21.45 (±0.76)
	Acc.	96.42 (±0.14)	93.41 (±0.18)	90.92 (±0.20)	90.05 (±0.20)	91.75 (±0.18)
$model_I$	JGA	41.17 (±2.08)	25.34 (±0.92)	17.57 (±0.48)	16.76 (±0.52)	21.11 (±0.65)
	Acc.	96.37 (±0.17)	93.29 (±0.22)	90.78 (±0.26)	89.91 (±0.27)	91.62 (±0.24)
$model_{SI}$	JGA	39.67 (±2.62)	27.70 (±1.32)	17.28 (±0.69)	16.51 (±0.59)	20.66 (±0.94)
	Acc.	96.16 (±0.29)	93.18 (±0.26)	90.68 (±0.26)	89.79 (±0.26)	91.51 (±0.25)
$model_{PK}$	JGA	46.39 (±2.27)	47.97 (±1.43)	38.96 (±0.86)	32.04 (±1.47)	40.07 (±0.90)
	Acc.	96.98 (±0.13)	96.87 (±0.10)	95.50 (±0.14)	94.55 (±0.22)	95.72 (±0.13)
$model_{AK}$	JGA	48.36 (±1.41)	50.36 (±1.19)	48.88 (±0.67)	47.20 (±0.78)	48.80 (±0.68)
	Acc.	97.13 (±0.10)	97.12 (±0.06)	96.73 (±0.05)	96.44 (±0.07)	96.79 (±0.05)

TABLEAU 6.4 – Joint Goal Accuracy (JGA) et slot accuracy (Acc.) moyennées sur 11 graines aléatoires différentes de différents modèles (voir section 6.5.1). Les écarts type sont donnés entre parenthèses.

		$test_K$	$test_{K+T}$	$test_{T+U}$	$test_U$	$test_{K+T+U}$
with	JGA	41.79	25.21	17.42	16.89	21.11
	Acc.	96.43	93.43	91.04	90.25	91.86
without	JGA	41.79	25.18	17.32	16.75	21.03
	Acc.	96.43	93.39	90.99	90.18	91.81

TABLEAU 6.5 – Effet de la mise à jour de l'ontologie avant l'évaluation sur les performances de $model_I$ sur différents jeux de données de *test*. Les performances rapportées correspondent à la Joint Goal Accuracy (JGA) et la slot accuracy (Acc.) de $model_I$ entraîné sur la graine aléatoire 100. L'ontologie est mise à jour avec l'ontologie correspondant au jeu de données de *test*.

traînement de chacun de ces trois modèles. Nous observons que les performances de $model_{PK}$ sont comprises entre les performances de $model_I$ et $model_{AK}$, de telle sorte que $model_{PK}$ et $model_{AK}$ obtiennent respectivement 5,22 et 7,19 points de plus de JGA que $model_I$. Dans la section 6.5.2, nous avions estimé que l'écart de JGA sur $test_K$ entre $model_I$ et $model_{AK}$ sur $test_K$ à cause de l'absence de certaines paires de $ontology_K$ dans les données d'entraînement de $model_I$ devait être de maximum 3,6 points. Cependant, comme l'écart observé est supérieur à cette estimation, nous pouvons en conclure que $model_I$ est significativement pénalisé par rapport aux autres modèles, à cause du fait qu'il est entraîné sur moins de données contenant des paires apparaissant dans $test_K$ que les autres modèles.

Capacités de généralisation

Nous répondons ici à la question Q1. Pour ce faire, nous comparons d'abord les performances de $model_I$ sur $test_K$, contenant des paires de slots vues pendant son entraînement, avec les performances du même modèle sur $test_{T+U}$, contenant des paires de slots jamais vues pendant l'entraînement. Nous observons alors une diminution des performances de 23,6 points de JGA. De la même manière, nous pouvons noter que les performances de $model_I$ sont bien inférieures à celles de $model_{AK}$ sur $test_{T+U}$ avec un écart de 31,31 points de JGA. Nous observons notamment que $model_I$ est capable de prédire correctement en moyenne seulement 22 tours sur les 4 471 tours de *test* (0,49%) qui contiennent des paires de $ontology_{T+U}$. Nous constatons donc que SUMBT n'est pas en mesure de généraliser sur des paires qu'il n'a jamais vues, ni dans ses données d'entraînement, ni dans l'ontologie utilisée pendant l'entraînement et que la connaissance de ces paires via l'ontologie pendant l'évaluation ne suffit pas. Ces observation sont similaires à celles faites par [Qian et al., 2021].

Pour aller plus loin, nous avons cherché à savoir si la mise à jour de l'ontologie avec les paires de $ontology_{T+U}$ avant l'évaluation sur $test_{T+U}$ avait un impact sur les performances de $model_I$. Les performances de $model_I$ entraîné sur une graine aléatoire en particulier, avec et sans mise à jour de l'ontologie, sont données dans la table 6.5. Ces données montrent que la mise à jour de l'ontologie semble n'avoir aucun impact sur les prédictions de $model_I$.

Capacités de transfert entre slots

Afin de répondre à la question Q2, nous comparons les performances de $model_I$, $model_{PK}$ et $model_{AK}$ sur $test_U$. Nous supposons que $model_{PK}$ est avantagé par rapport à $model_I$ puisqu'il a déjà connaissance de l'ensemble des valeurs de slots de $test_U$ même si $test_U$ contient des paires qui lui sont inconnues. Nous observons que les performances de $model_{PK}$ sont comprises entre les performances de $model_I$ et $model_{AK}$, de telle sorte que $model_{PK}$ obtient 15,28 points de plus que $model_I$ et que $model_{AK}$ obtient 15,16 points de plus que $model_{PK}$. Si nous comparons $model_{PK}$ et $model_I$, nous observons que $model_{PK}$ est capable de prédire correctement - au contraire de $model_I$ - en moyenne 311/2 222 tours de $test_U$, soit 14,02% des tours de $test_U$, qui contiennent des paires de $ontology_U$. Nous pouvons donc conclure que, le fait de connaître pendant l'entraînement des valeurs de $test_U$ sans connaître les paires elles-même, est bénéfique pour $model_{PK}$, c'est-à-dire qu'il existe un transfert entre types de slots lorsque des valeurs sont communes à plusieurs types de slots.

Cependant, ce transfert ne semble pas optimal. En effet, en section 6.5.2 nous avions estimé que si SUMBT n'était pas en mesure de généraliser sur des nouvelles

paires de slots, l'écart entre $model_{PK}$ et $model_{AK}$ sur $test_U$ serait de minimum 5,97%. De plus, si un écart de performance peut aussi être expliqué par le fait que chaque modèle n'est pas entraîné sur le même nombre de données, nous pouvons estimer cet écart à seulement environ 2 point de JGA comme vu en début de cette section. Ainsi les 7 points de JGA restants s'expliquent par le fait que $model_{PK}$ n'est pas en mesure de profiter pleinement du transfert de connaissances entre slots ayant des valeurs en commun.

Impact de l'ontologie pendant l'entraînement

Afin de voir si l'ontologie utilisée pendant l'entraînement a un impact sur les performances sur $test_{T+U}$, nous comparons les performances de $model_{2I}$, $model_{EI}$, $model_I$ et $model_{SI}$ sur ce jeu de données de test. Nous pouvons observer que les performances de tous ces modèles sont équivalentes puisque les performances de chaque modèle sont comprises dans les écart-type de chaque modèle. Cette observation renforce l'idée que SUMBT nécessite de connaître les paires de slots dans le cadre de dialogue et que la connaissance via l'ontologie ne lui suffit pas pour prédire ces slots.

6.5.4 Discussion

Dans cette section nous avons prouvé de manière empirique que SUMBT n'est pas en mesure de généraliser sur des paires de slots qu'il n'a jamais vues et dont les valeurs n'ont jamais été rencontrées avec d'autres types de slots au cours de son entraînement. Les observations mettent en particulier en lumière la nécessité pour SUMBT de voir les paires de slots *en contexte*, c'est-à-dire dans le cadre de dialogues, puisque la connaissance via son ontologie de ces paires durant son entraînement ou juste pendant l'évaluation ne suffit pas pour les prédire correctement. Cependant, nous avons aussi montré de manière empirique que SUMBT permet le transfert, non optimal mais significatif, de connaissances entre slots ayant des valeurs en commun. Ceci est un point positif de SUMBT, d'autant plus qu'en pratique il n'est pas rare que des slots partagent des valeurs communes, au sein d'un même domaine comme entre différents domaines.

Notre étude présente cependant certaines limites. En effet, les observations faites dans cette section s'appliquent à des données préparées avec une graine aléatoire spécifique, et il serait intéressant de préparer des données avec une autre graine aléatoire afin de vérifier à quel point nos observations sont généralisables. De plus, notre configuration, bien que permettant l'analyse du transfert entre types de slots ayant des valeurs de slot en commun, implique par exemple de comparer les performances de $model_I$ sur $test_K$ et $test_{T+U}$ alors que ces deux jeux de données

ne sont pas forcément comparables, puisque les dialogues dans le sous ensemble *known* peuvent être plus complexes que dans $transfert + unknown$, ou inversement. Notre configuration implique aussi de comparer différents domaines qui ne sont pas entraînés sur le même nombre de données.

Cependant, malgré ces limites et grâce à l'analyse des données, nous avons observé que SUMBT est capable de transférer des connaissances d'un type de slot à un autre. Il est intéressant d'étudier si ces observations sont toujours valables dans un cadre différent où le modèle doit prédire non plus des valeurs de slot inconnues mais des types de slots inconnus. De plus cette configuration se rapproche plus du LL.

6.6 Prédire des types slots relatifs à de nouveaux domaines

Une caractéristique hautement souhaitable des systèmes de dialogue est la capacité à s'adapter à de nouveaux domaines sans entraînement supplémentaire, mais en tirant parti des connaissances déjà acquises dans les domaines précédents. C'est pourquoi nous étudions dans cette section le transfert entre domaines de type « leave-one-out ». Pour chaque domaine, un modèle est entraîné sur des dialogues ne contenant pas de slots du domaine cible et est ensuite évalué sur des dialogues contenant les slots du domaine cible. Dans cette section nous cherchons en particulier à améliorer les performances zero-shot de SUMBT [Lee et al., 2019]. Pour cela nous proposons plusieurs variantes de l'architecture originale de SUMBT ainsi qu'une méthode applicable pendant l'inférence.

En section 6.6.1 nous présentons en détail la configuration des expériences de cette section. Ensuite, en section 6.6.2 nous présentons les résultats du transfert zero-shot sur un nouveau domaine de SUMBT. Afin d'améliorer ces résultats, nous proposons en section 6.6.3 une méthode applicable pendant l'inférence, appelée *modulation de l'attention*, ainsi que plusieurs variantes de SUMBT, comme présenté en section 6.6.4. En section 6.6.5, nous présentons ensuite le transfert zero-shot sur chaque domaine de nos variantes, auxquelles nous avons appliqué la modulation de l'attention. Enfin, en section 6.6.6 nous discutons les résultats obtenus précédemment.

La majorité des expériences décrites dans cette section a fait l'objet d'un article présenté au workshop COLING CODI (Computational Approaches to Discourse) de 2022 [Veron et al., 2022]. Le code associé à l'article est mis à disposition afin de permettre la reproduction des expériences : https://github.com/mathilde-veron/attention-modulation-zero-dst.

6.6.1 Configuration des expériences

Conformément aux autres travaux qui s'intéressent au transfert entre domaines dans le cadre de l'apprentissage zero-shot, le transfert est mesuré en calculant la Joint Goal Accuracy (JGA) uniquement sur les slots du domaine cible. Ainsi, la JGA utilisée pour les expériences zero-shot de cette section correspond au pourcentage de tours sur l'ensemble des dialogues, où tous les slots *du domaine cible* ont correctement été prédits. En conséquence, la JGA sur un seul domaine sera plus élevée que la JGA sur l'ensemble des domaines. Toutes les expériences sont effectuées sur cinq graines aléatoires.

Afin de comparer nos performances zero-shot avec l'état de l'art, nous réalisons nos expériences sur MultiWOZ 2.0 [Budzianowski et al., 2018a] et MultiWOZ 2.1 [Eric et al., 2020]. Pour les expériences sur MultiWOZ 2.0, comme précédemment, les données et l'ontologie utilisées sont celles fournies par les auteurs de SUMBT[3]. À noter que l'ontologie a été modifiée par les auteurs afin de permettre au modèle SUMBT de l'utiliser. Pour les expériences sur MultiWOZ 2.1, les données originales [4] ont été formatées en utilisant les scripts fournis par les auteurs de SUMBT. Cependant, comme aucune ontologie n'était disponible [5] pour cette version nous avons dû extraire l'ontologie des données et nettoyer nous-même l'ontologie afin qu'elle puisse être utilisée par SUMBT. Le nettoyage consistait à supprimer les doublons, comme par exemple la destination en train « London Liverpool Street » qui apparaissait à la fois sous la valeur « london liverpool street » et la valeur « liverpool street » ou comme les restaurants apparaissant à la fois avec et sans le déterminant « the ». Le résultats de ce nettoyage en termes de nombre de valeurs par type de slot est présenté dans le tableau 6.7.

Nous répétons les expériences sur les cinq domaines de MultiWOZ qui disposent de données de test : `attraction`, `hotel`, `restaurant`, `taxi` et `train`. La quantité de données utilisée pour les expériences sur chaque domaine cible est donné dans le tableau 6.6.

Pour réaliser ces expériences nous avons ré-implémenté le modèle SUMBT à partir de l'article original et du code mis à disposition par les auteurs. La ré-implémentation du modèle nous a permis de faciliter la modification de l'architecture originale nécessaire à la création des variantes ainsi que l'implémentation de notre méthode.

4. Voir https://github.com/budzianowski/multiwoz.
5. Une ontologie était disponible mais celle-ci était directement extraite des données MultiWOZ 2.1 ce qui n'est pas réaliste.

Domaine cible	# Dialogues			# tours		
	train	*dev*	*test*	*train*	*dev*	*test*
attraction	5 758	596	398	36 206	4 260	3 135
hotel	5 063	582	396	29 909	4 003	3 225
restaurant	4 583	552	448	29 434	3 904	3 433
taxi	6 968	790	198	44 640	5 714	1 573
train	5 473	518	491	34 052	3 660	3 808

TABLEAU 6.6 – Nombre de données pour chaque domaine cible pour les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.0.

6.6.2 Expériences préliminaires sur SUMBT

Dans un premier temps, nous réalisons les expériences zero-shot sur SUMBT original afin d'analyser ses performances et de permettre par la suite la comparaison des performances originales avec celles obtenues dans le cadre de nos expériences.

Les résultats sur MultiWOZ 2.0 et 2.1 sont présentés dans le tableau 6.8 [6]. Nous pouvons observer que SUMBT a de mauvaises performances zero-shot, alors que son ontologie est mise à jour avant l'évaluation avec la liste des valeurs de chaque type de slot du domaine cible. Ces résultats ne sont pas étonnants étant donné que l'on avait observé en section 6.5 que SUMBT n'était pas capable de généraliser sur de nouvelles valeurs de slots, pourtant présentes pendant l'évaluation dans l'ontologie.

En examinant de plus près les prédictions du modèle, nous remarquons que SUMBT a tendance à prédire la valeur de slot `none` plus qu'il ne le devrait. En effet, la proportion de valeurs `none` dans les données d'entraînement est en moyenne de 71%, alors que le modèle prédit en moyenne 78% du temps la valeur `none` sur les données de test des domaines utilisés pendant l'entraînement. Lorsque l'on applique le modèle à des domaines inconnus, la proportion augmente en moyenne jusqu'à 88% et peut même atteindre 99% dans le cas du domaine `attraction`. Nous constatons donc que cette tendance s'intensifie lorsqu'un nouveau type de slot jamais vu lors de l'entraînement est passé comme requête en entrée du modèle.

6. Nous ne savons pas d'où vient l'écart de performance observé sur le domaine `hotel` entre nos expériences et celles de [Campagna et al., 2020]. Nous supposons que ceci est dû à l'initialisation de leur modèle qui devait être avantageuse, puisqu'il nous semble que leurs expériences n'ont pas été répétées sur plusieurs graines aléatoires.

Type de slot	**Ontologie**		
	MultiWOZ 2.0	**MultiWOZ 2.1**	**MultiWOZ 2.1 (nettoyée)**
attraction-area	6	6	5
attraction-name	144	163	126
attraction-type	20	32	24
hotel-area	13	6	5
hotel-book day	8	12	7
hotel-book people	8	8	8
hotel-book stay	8	10	8
hotel-internet	2	3	3
hotel-name	91	88	61
hotel-parking	3	3	3
hotel-price range	3	7	4
hotel-stars	6	8	6
hotel-type	2	4	3
restaurant-area	5	6	5
restaurant-book day	9	9	7
restaurant-book people	8	9	8
restaurant-book time	60	71	61
restaurant-food	103	108	97
restaurant-name	190	189	156
restaurant-price range	3	4	3
taxi-arrive by	97	96	96
taxi-departure	271	254	246
taxi-destination	282	252	250
taxi-leave at	118	107	107
train-arrive by	109	156	154
train-book people	12	11	11
train-day	11	7	7
train-departure	39	30	29
train-destination	32	26	25
train-leave at	146	202	197

TABLEAU 6.7 – Nombre de valeurs par types de slots et par ontologie (hors `none` et `don't care`). L'ontologie de MultiWOZ 2.1 non nettoyée a été extraite des données MultiWOZ 2.1. L'ontologie de MultiWOZ 2.1 nettoyée correspond à l'ontologie extraite des données MultiWOZ 2.1 que nous avons nettoyé nous-même.

Implémentation	Données	Attraction	Hotel	Restaurant	Taxi	Train
La nôtre	2.0	23.57 ±0.86	14.51 ±1.23	17.19 ±0.84	60.41 ±0.12	21.31 ±0.91
La nôtre	2.1	22.80 ±0.52	14.02 ±1.86	16.57 ±0.92	58.96 ±1.13	21.59 ±0.60
[Lee et al., 2019]∗	2.1	22.6	19.8	16.5	59.5	22.5

TABLEAU 6.8 – Résultats des expériences de transfert zero-shot entre domaines de différentes implémentations de SUMBT sur différentes versions de MultiWOZ. Les domaines en colonne indiquent le domaine cible considéré. ∗Résultats rapportés par [Campagna et al., 2020].

6.6.3 Modulation de l'attention

Motivés par les observations précédentes, nous proposons une méthode appelée *modulation de l'attention*. Cette méthode a pour but d'inciter le modèle à moins prédire la valeur de slot `none` pour les types de slots inconnus. Plus précisément, le but est d'appliquer cette méthode au modèle, uniquement lorsque le tour de dialogue, dont l'état doit être prédit, fait référence à un domaine inconnu.

Nous avons vu en section 6.4 que SUMBT s'appuie sur une couche d'attention multi-tête (voir 2.1.2). Celle-ci permet à SUMBT d'attirer son attention sur les tokens liés au type de slot passé comme requête en entrée. À des fins d'illustration, les explications données ici sont appliquées uniquement à une tête d'attention et les dimensions dans les équations ne tiennent pas compte du nombre de têtes. Pour rappel, le mécanisme d'attention prend en entrée trois matrices : Q un ensemble de requêtes, K un ensemble de clés, et V un ensemble de valeurs. Dans notre cas, nous avons $Q \in \mathbb{R}^{1\times d}$, où Q correspond à q^s la paire domaine/nom de slot encodée par $BERT_{sv}$ et d dénote la dimension du modèle BERT. $K \in \mathbb{R}^{sl\times d}$ et $V \in \mathbb{R}^{sl\times d}$ correspondent tous deux à la concaténation d'un énoncé du système et d'un énoncé de l'utilisateur (un tour de dialogue) codé par $BERT$ également noté $U_t = \{u_{t,i}\}_{i\in[\![0,sl]\!]}$, où t désigne un indice de tour unique sur tous les dialogues et sl le nombre maximum de tokens pouvant être encodés par $BERT$, y compris les tokens spéciaux `[CLS]` et `[SEP]`. Le mécanisme d'attention réalisé dans SUMBT est formalisé comme suit :

$$Attention(Q, K, V) = (w_{t,i}^{d^s}) \cdot V \tag{6.1}$$

$$\text{avec } (w_{t,i}^{d^s})_{i\in[\![0,sl]\!]} = softmax\left(\frac{s_{t,i}^{d^s}}{\sqrt{d}}\right) \tag{6.2}$$

$$\text{et } (s_{t,i}^{d^s})_{i\in[\![0,sl]\!]} = Q \cdot K^T \tag{6.3}$$

Où d^s désigne le domaine associé au type de slot s et $w_{t,i}^{d^s}$ correspond aux poids d'attention appliqués à U_t (la matrice des valeurs V) après normalisation des scores d'attention $s_{t,i}^{d^s}$.

Dans leur article, les auteurs de SUMBT ont découvert que les poids d'attention étaient élevés sur les tokens spéciaux `[CLS]` et `[SEP]` lorsque la slot-value `none` était prédite. Pour pousser le modèle à prédire des valeurs autres que la valeur `none`, nous pouvons alors simplement réduire les poids d'attention sur ces tokens spéciaux. Nous appelons cette méthode *modulation de l'attention* et la définissons

comme suit :

$$(w_{t,i}^{d^s})_{i\in[\![0,sl]\!]} = softmax\left(\frac{\alpha_{t,i}^{d^s}\cdot s_{t,i}^{d^s}}{\sqrt{d}}\right) \tag{6.4}$$

$$\text{avec } \alpha_{t,i}^{d^s} = \begin{cases} 0 & \text{si } d^s \in ND \text{ et } u_{t,i} \in ST, \\ 1 & \text{sinon.} \end{cases} \tag{6.5}$$

Où *ND* est l'ensemble des nouveaux domaines jamais vus pendant l'entraînement, et *ST* est l'ensemble des tokens spéciaux `[CLS]` et `[SEP]`. Cette méthode est simple et avantageuse car elle ne nécessite pas d'apprentissage supplémentaire et peut être appliquée directement au modèle pendant l'inférence.

6.6.4 Variantes de SUMBT

Dans la section précédente, nous avons décrit une méthode visant à moduler l'attention du modèle sur les tokens spéciaux, afin de pousser le modèle à moins prédire la valeur `none`. Cependant, cela pourrait conduire le modèle à prédire une valeur inexacte. Ainsi, nous décrivons dans cette section trois variantes de SUMBT, visant à tirer parti des similitudes qui existent naturellement entre les slots des différents domaines. Nous faisons l'hypothèse que cela pourrait aider le modèle à augmenter le transfert entre les domaines et que notre méthode pourrait être plus efficace sur ces variantes.

En effet, certains slots peuvent partager le même nom (*e.g.* `destination` ou `departure` pour les domaines train et taxi), le même type de valeurs (*e.g.* les valeurs de `hotel-stars` et `restaurant book people` correspondent tous deux à des nombres) ou même les mêmes valeurs (*e.g.* `the gardenia` est aussi bien un nom de restaurant qu'une destination de taxi). Pour stimuler le transfert entre domaines, nous décrivons chaque type de slot avec son domaine, son nom, et le type de ses valeurs, suivant les descriptions « slot type » décrites par [Lin et al., 2021b]. Nous supposons que les variantes de SUMBT incorporant ces descriptions pourront bénéficier davantage de la modulation de l'attention que le modèle original. Nous proposons donc trois variantes principales de SUMBT :

— **Avec description des types de slots** : L'architecture du modèle est la même que pour SUMBT original, cependant nous utilisons directement les descriptions des types de slots en langage naturel « slot type » [Lin et al., 2021b] à la place de `<domain>-<slot name>`. Par exemple, au lieu de `"[CLS] hotel-book people [SEP]"`, le modèle prend en entrée la requête correspondant au texte `"[CLS] Number of people for the hotel booking [SEP]"`.

— **Avec triple requête** (Figure 6.3a) : La requête q^s consiste ici en une matrice de 3 vecteurs correspondant respectivement au nom, au type de valeurs, et au domaine du type de slot sur lequel porte la requête, les trois étant encodés par $BERT_{sv}$. Puisque nous avons maintenant $q^s \in \mathbb{R}^{3\times d}$, la couche d'attention multi-tête donne en sortie $g_t^s \in \mathbb{R}^{3\times d}$. Pour récupérer une sortie aux bonnes dimensions (h_t^s), nous concaténons d'abord les trois vecteurs puis appliquons une couche linéaire $h_t^s = g_t^s W + b$ avec $W \in \mathbb{R}^{3*d\times d}$ suivie d'une activation ReLU [Nair and Hinton, 2010].

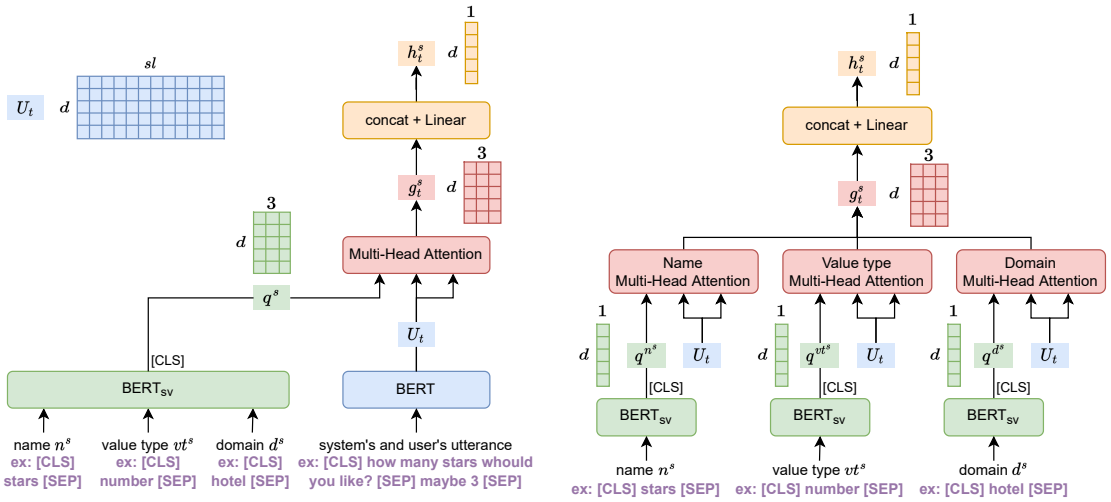
— **Avec triple attention** (Figure 6.3b) : Nous utilisons 3 couches d'attention multi-têtes indépendantes et donnons en entrée de chaque couche respectivement le nom, le type de valeurs, et le domaine du type de slot sur lequel porte la requête, les trois étant codés par $BERT_{sv}$. Les sorties de chaque couche d'attention multi-tête sont ensuite concaténées, et le vecteur résultant est transformé de la même manière que précédemment afin d'obtenir le vecteur h_t^s. Nous supposons qu'entraîner le modèle à porter son attention sur ces trois aspects de manière indépendante l'aidera à généraliser et ainsi favorisera le transfert vers de nouveaux slots.

Pour ces trois variantes ainsi que pour le modèle SUMBT original, nous ajoutons également des variantes où les poids de l'encodeur des tours de dialogue $BERT$ sont fixés pendant l'entraînement. Nous supposons que cela pourrait aider le modèle à généraliser sur des domaines inconnus. La fixation de ses poids a également l'avantage de réduire considérablement le coût de calcul de chaque époque d'apprentissage.

6.6.5 Expériences et résultats

Dans ces expériences, nous avons utilisé un oracle pour détecter le domaine associé au tour de dialogue en entrée. Le domaine associé à chaque tour de dialogue a été extrait des données MultiWOZ 2.1 à partir des actes de dialogue de l'utilisateur.

La modulation de l'attention a été pensée pour être appliquée uniquement à l'attention liée au domaine, c'est-à-dire sur la requête ou la couche d'attention spécifique au domaine, respectivement pour la variante *avec triple requête* et la variante *avec triple attention*. En effet la modulation se faisant sur les tours de dialogue qui sont associés à un nouveau domaine, il paraît donc logique d'appliquer la modulation uniquement à l'attention qui se réfère au domaine. Pour valider cette hypothèse, nous évaluons aussi le modèle en appliquant la modulation respectivement sur l'ensemble des 3 requêtes et sur l'ensemble des trois couches d'attention. Pour SUMBT original et la variante *avec description des types de slots*, comme il n'y a pas d'attention spécifique au domaine, ces considérations ne sont pas applicables et la modulation est appliquée à la couche d'attention du type de slot.



(a) avec triple requête (b) avec triple attention

FIGURE 6.3 – Variantes de SUMBT [Lee et al., 2019]. Après h_t^s l'architecture reste la même que le modèle original comme illustré en figure 6.2. Les dimensions d et sl correspondent respectivement à la dimension de $BERT$ et au nombre maximum de tokens qui peuvent être encodés par $BERT$.

Résultats sans modulation

Les résultats sur MultiWOZ 2.0 sont présentés dans le tableau 6.9. De manière générale, il semble que les variantes proposées n'améliorent pas le transfert entre domaines. Il n'est en effet pas possible d'identifier une variante pour laquelle les performances zero-shot sont améliorées par rapport à celles de SUMBT original sur tous les domaines. Dans le domaine `attraction`, les résultats des différentes variantes sont similaires aux résultats originaux de SUMBT [7]. Dans les domaines `hotel` et `train`, toutes les variantes sont plus performantes que l'original. Cependant, dans les domaines `restaurant` et `taxi`, presque toutes les variantes sont moins performantes que l'original. [8] Nous observons également que le fait de fixer les poids de $BERT$ pendant l'apprentissage aide les variantes correspondantes à obtenir de meilleures performances seulement dans la moitié des cas, par rapport aux variantes où $BERT$ est adapté. Nous ne pouvons donc pas affirmer que fixer les poids de $BERT$ est bénéfique pour le transfert. En particulier, sur le domaine `restaurant`, les variantes dont les poids de $BERT$ sont fixés semblent être parti-

7. Mis à part la variante *avec description des types de slots* dont la performance est inférieure aux autres de plus de deux points de JGA, et la variante *avec triple requête*, dont les poids de $BERT$ sont fixés, qui obtient presque un point de JGA de plus que les autres variantes, même si cet écart n'est pas significatif si l'on considère l'écart type de cette performance.

8. À l'exception de la variante *avec triple requête* dans le domaine `restaurant`.

Version	Attraction	Hotel	Restaurant	Taxi	Train
Original	23.57 ±0.86	<u>14.51</u> ±1.23	17.19 ±0.84	**60.41** ±0.12	<u>21.31</u> ±0.91
+ frozen BERT	23.29 ±0.25	15.09 ±0.31	14.94 ±1.26	60.29 ±0.17	22.61 ±0.18
w/ slot desc. queries	<u>21.17</u> ±1.67	**16.67** ±0.98	<u>14.82</u> ±0.56	60.25 ±0.50	21.69 ±0.09
w/ triple queries	23.56 ±2.09	16.02 ±1.17	**18.16** ±1.19	<u>56.11</u> ±3.60	21.42 ±1.59
+ frozen BERT	**24.52** ±1.07	15.92 ±0.78	15.58 ±0.32	58.17 ±1.75	22.61 ±0.33
w/ triple attention	23.70 ±4.51	16.06 ±0.90	16.41 ±2.46	56.88 ±3.31	22.54 ±0.32
+ frozen BERT	23.32 ±1.64	15.55 ±0.90	15.65 ±1.20	59.68 ±0.83	**22.74** ±0.07

TABLEAU 6.9 – JGA de différentes variantes de SUMBT sur les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.0, sans modulation de l'attention durant l'évaluation. Les colonnes désignent le domaine cible et le signe ± indique l'écart type. La meilleure performance pour chaque domaine cible est indiquée en gras, tandis que la moins bonne est soulignée. « w/ slot desc. queries » correspond à la variante *avec description des types de slots*, « w/ triple queries » correspond à la variante *avec triple requête*, « w/ triple attention » correspond à la variante *avec triple attention* et « + frozen BERT » indique les variantes dont les poids de *BERT* sont fixés.

culièrement désavantagées puisque leurs performances sont systématiquement plus faibles que les variantes correspondantes où *BERT* est adapté. À l'inverse, sur le domaine `taxi` les variantes *avec triple requête* et *avec triple attention*, dont les poids de *BERT* sont fixés, gagnent au minimum deux points de JGA par rapport à leur variante de base. À noter que dans l'ensemble, fixer les poids de *BERT* donne moins de variation sur les résultats.

Les résultats sur MultiWOZ 2.1 sont présentés dans le tableau 6.10. Les résultats sont assez similaires à ceux sur MultiWOZ 2.0 et de la même manière nous n'observons pas d'apport général grâce aux variantes.

Résultats avec modulation

Les résultats sur MultiWOZ 2.0 sont présentés dans le tableau 6.11. Dans un premier temps, nous observons dans la totalité des cas que les performances sont meilleures lorsque la modulation est appliquée uniquement à l'attention associée au domaine, et non aux attentions associées au nom, au type de valeurs, et au domaine (dans le tableau nous comparons pour la variante « w/ triple queries » la modulation « on domain query » avec la modulation « on all queries » et pour la variante « w/ triple attn. » la modulation « on domain attn. » avec la modulation « on all attn. »). Ceci valide notre hypothèse de départ qui stipulait que seule l'attention liée au domaine devait être modulée. Par la suite, lorsque nous parlons de modulation nous faisons toujours référence à la modulation sur l'attention liée

Version	Attraction	Hotel	Restaurant	Taxi	Train
Original	**22.75** ±0.46	14.02 ±1.86	16.57 ±0.92	58.96 ±1.13	21.59 ±0.60
+ frozen BERT	22.52 ±0.05	14.04 ±0.51	14.30 ±0.76	**59.28** ±0.12	22.06 ±0.33
w/ triple queries	21.85 ±2.64	16.61 ±1.18	16.85 ±0.69	54.63 ±3.23	21.96 ±1.15
+ frozen BERT	21.29 ±1.76	15.13 ±0.74	15.82 ±0.57	57.27 ±1.63	21.79 ±1.10
w/ triple attention	21.55 ±1.55	**16.64** ±0.42	**17.10** ±1.26	52.96 ±3.66	21.98 ±1.38
+ frozen BERT	22.01 ±1.03	14.41 ±0.84	13.84 ±0.88	58.36 ±0.98	**22.29** ±0.27

TABLEAU 6.10 – JGA de différentes variantes de SUMBT sur les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.1, sans modulation de l'attention durant l'évaluation. Les informations supplémentaires données dans le tableau 6.9 s'appliquent ici aussi.

au domaine pour les variantes *avec triple requête* et *avec triple attention*.

Ensuite, toujours sur MultiWOZ 2.0, nous observons que la variante *avec triple attention*, dont les poids de *BERT* sont fixés et à laquelle la modulation a été appliquée, obtient les meilleurs résultats sur les domaine `attraction` et `hotel` avec une augmentation importante de respectivement 6,26 et 2,58 points par rapport à SUMBT original sans modulation. Dans le domaine `restaurant`, la variante *avec triple requête*, dont les poids de *BERT* sont fixés et à laquelle la modulation a été appliquée, obtient les meilleurs résultats avec une augmentation de 1,94 points par rapport à SUMBT original sans modulation. Cependant, la modulation ne semble pas avoir d'impact sur les domaines `taxi` et `train`.

Les performances sur MultiWOZ 2.1 sont un peu différentes comme présenté dans le tableau 6.12. Sur le domaine `attraction`, la meilleure performance est toujours réalisée par la variante *avec triple attention*, dont les poids de *BERT* sont fixés, avec modulation, cependant le gain de performance par rapport à MultiWOZ 2.0 est moins élevé et atteint +3,81 points de JGA. Sur les domaines `hotel` et `restaurant` la variante *avec triple attention* obtient les meilleures performances avec un gain de respectivement +3,19 et +1,54 points. Pour les domaines `taxi` et `train`, la modulation ne semble pas avoir d'impact de la même manière que pour MultiWOZ 2.0.

Afin de mieux observer l'apport effectif de la modulation, nous calculons pour chaque modèle entraîné sur une graine aléatoire spécifique la différence de ses performances avec et sans modulation. Les différences résultant de ce calcul sont moyennées sur l'ensemble des graines aléatoires par variantes et par domaines. Ces moyennes correspondent à la troisième ligne (et à la cinquième ligne si présente) de chaque variante dans les tableaux 6.11 et 6.10. De manière générale, nous constatons que la modulation permet d'augmenter les performances. En effet, la

différence est presque toujours positive, et si ce n'est pas le cas, elle est contenue dans l'écart-type ou proche de celui-ci. Dans les domaines `attraction` et `hotel`, les variantes *avec triple attention* bénéficient davantage de la modulation que les variantes *avec triple requête*. Sur MultiWOZ 2.1 cette remarque est aussi valable sur le domaine `restaurant`. Cela suggère que le fait que le nom, le type de valeur et le domaine du type de slot passé en entrée possèdent leur propre mécanisme d'attention est plus bénéfique pour le transfert. Plus précisément, sur ces trois domaines, la variante *avec triple attention* dont les poids de $BERT$ sont fixés est celle qui bénéficie le plus de la modulation avec une augmentation de respectivement +6,51, +1,54 et +1,15 sur MultiWOZ 2.0 et de respectivement +4,56, +1,52 et +1,36 sur MultiWOZ 2.1. De manière surprenante, nous observons que la modulation fonctionne sur SUMBT original dont les poids de $BERT$ sont fixés dans les domaines `attraction` et `restaurant`. En revanche, dans les domaines `taxi` et `train`, la modulation a un impact négligeable sur les performances de toutes les variantes. En dehors de ces deux domaines, la modulation semble avoir un meilleur impact lorsque les poids de $BERT$ sont fixés pendant l'entraînement (deux tiers du temps).

6.6.6 Discussion

Comment expliquer les différences entre les résultats sur MultiWOZ 2.0 et 2.1 ?

Nous avons pu observer certaines différences entre les performances sur MultiWOZ 2.0 et 2.1. Ainsi dans le tableau 6.8 nous avons vu que le modèle SUMBT original est moins performant sur MultiWOZ 2.1 sur la majorité des domaines. Cet effet peut être dû au fait que l'ontologie que nous avons utilisée pour nos expériences sur MultiWOZ 2.1 a été nettoyée notamment afin de supprimer les doublons (voir tableau 6.7). SUMBT a été originellement pensé pour gérer les variations d'une même entité comme mis en avant par ses concepteurs, cependant cette capacité n'a jamais été vérifiée. Il est donc possible que SUMBT tende à mémoriser par cœur les valeurs et qu'il ne soit de fait pas aidé par le nettoyage de l'ontologie. Il serait utile de vérifier cette hypothèse en renouvelant les expériences sur MultiWOZ 2.1 mais avec l'ontologie directement extraite des jeux de données *train*, *dev* et *test*.

Il est aussi possible que les corrections réalisées pour la version 2.1 mènent à des annotations plus pointues et plus difficiles à prédire pour SUMBT. En effet [Eric et al., 2020] ont observé une baisse de performance similaire sur d'autres modèles et ont constaté qu'un grand nombre d'erreurs était dû à la difficulté du modèle étudié à appréhender la valeur `don't care`.

Version	Modulation	Attraction	Hotel	Restaurant	Taxi	Train
Original	none	23.57 ±0.86	14.51 ±1.23	17.19 ±0.84	60.41 ±0.12	21.31 ±0.91
	on slot attn.	25.03 ±3.04	14.23 ±1.07	17.81 ±1.07	**60.48** ±0.15	21.25 ±0.88
		+1.46 ±2.19	−0.28 ±0.24	+0.62 ±1.46	+0.08 ±0.11	−0.06 ±0.05
+ frozen BERT	none	23.29 ±0.25	15.09 ±0.31	14.94 ±1.26	60.29 ±0.17	22.61 ±0.18
	on slot attn.	28.00 ±1.06	15.62 ±0.48	17.30 ±0.88	60.28 ±0.17	22.62 ±0.19
		+4.71 ±1.02	+0.53 ±0.22	+2.36 ±1.12	−0.01 ±0.03	+0.01 ±0.02
w/ triple query	none	23.56 ±2.09	16.02 ±1.17	18.16 ±1.19	56.11 ±3.60	21.42 ±1.59
	on domain query	25.40 ±1.78	16.14 ±0.95	**19.13** ±0.80	56.26 ±3.71	21.43 ±1.62
		+1.85 ±2.88	+0.12 ±0.41	+0.97 ±0.64	+0.15 ±0.26	+0.01 ±0.04
	on all queries	24.40 ±1.12	15.67 ±1.39	18.25 ±0.87	56.22 ±3.70	21.42 ±1.61
		+0.84 ±2.48	−0.35 ±0.67	+0.09 ±1.49	+0.12 ±0.21	−0.00 ±0.03
+ frozen BERT	none	24.52 ±1.07	15.92 ±0.78	15.58 ±0.32	58.17 ±1.75	22.61 ±0.33
	on domain query	25.58 ±1.36	15.90 ±0.70	16.99 ±0.58	58.13 ±1.77	22.63 ±0.31
		+1.06 ±1.23	−0.02 ±0.46	+1.40 ±0.68	−0.04 ±0.10	+0.02 ±0.03
	on all queries	25.19 ±1.38	15.68 ±0.76	16.85 ±0.48	58.22 ±1.79	22.64 ±0.27
		+0.67 ±1.05	−0.24 ±0.46	+1.27 ±0.63	+0.05 ±0.05	+0.03 ±0.07
w/ triple attn.	none	23.70 ±4.51	16.06 ±0.90	16.41 ±2.46	56.88 ±3.31	22.54 ±0.32
	on domain attn.	28.53 ±4.99	16.37 ±0.88	18.29 ±2.00	56.96 ±3.38	22.58 ±0.33
		+4.83 ±2.42	+0.31 ±0.09	+1.88 ±1.63	+0.08 ±0.08	+0.04 ±0.05
	on all attn.	26.88 ±4.35	16.00 ±0.83	17.14 ±2.19	56.94 ±3.39	22.63 ±0.30
		+3.18 ±1.95	−0.06 ±0.22	+0.74 ±0.66	+0.05 ±0.09	+0.08 ±0.12
+ frozen BERT	none	23.32 ±1.64	15.55 ±0.90	15.65 ±1.20	59.68 ±0.83	**22.74** ±0.07
	on domain attn.	**29.83** ±1.57	**17.09** ±1.37	16.80 ±1.30	59.72 ±0.84	**22.74** ±0.07
		+6.51 ±0.87	+1.54 ±0.68	+1.15 ±0.60	+0.04 ±0.06	−0.00 ±0.03
	on all attn.	28.39 ±1.23	16.76 ±1.44	16.27 ±1.22	59.76 ±0.84	**22.75** ±0.08
		+5.07 ±0.51	+1.21 ±0.73	+0.61 ±0.62	+0.08 ±0.07	+0.00 ±0.03

TABLEAU 6.11 – JGA de différentes variantes de SUMBT sur les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.0, avec et sans modulation de l'attention durant l'évaluation. Les informations supplémentaires données dans le tableau 6.9 s'appliquent ici aussi.

Observations sur les proportions de slots prédits avec la valeur `none`

Afin d'analyser les résultats plus en détail, nous présentons dans le tableau 6.13 la proportion de slots prédits avec la valeur `none` pour chaque variante sur MultiWOZ 2.0. Dans un premier temps, nous pouvons remarquer que les variantes dont les poids de *BERT* ont été fixés ont tendance à davantage prédire la valeur `none` par rapport aux autres variantes. Si l'on compare le tableau précédent avec le tableau 6.11, le fait qu'une variante prédise plus de `none` qu'une autre est associé dans 3/4 des cas à un gain plus important de la modulation lorsque l'on compare les mêmes variantes avec et sans les poids de *BERT* fixés. Cette observation semble en partie expliquer pourquoi les variantes dont les poids de *BERT* sont fixés bénéficient en majorité davantage de la modulation de l'attention.

Version	Modulation	Attraction	Hotel	Restaurant	Taxi	Train
Original	none	22.75 ±0.46	14.02 ±1.86	16.57 ±0.92	58.96 ±1.13	21.59 ±0.60
	on slot attn.	26.36 ±4.64	13.76 ±2.30	17.11 ±1.23	58.96±1.09	21.57±0.75
		+3.61 ±4.81	-0.26±0.44	+0.55±1.16	+0.00 ±0.05	-0.01 ±0.19
+ frozen BERT	none	22.52 ±0.05	14.04 ±0.51	14.30 ±0.76	**59.28** ±0.12	22.06 ±0.33
	on slot attn.	24.67 ±0.65	14.45 ±0.47	16.21 ±1.01	**59.28** ±0.12	22.07 ±0.36
		+2.16 ±0.63	+0.42 ±0.09	+1.92±0.76	+0.00±0.00	+0.01±0.05
w/ triple queries	none	21.85 ±2.64	16.61 ±1.18	16.85 ±0.69	54.63 ±3.23	21.96 ±1.15
	on domain query	23.74 ±3.14	15.95±1.39	17.54 ±0.99	54.79 ±3.20	21.93 ±1.22
		+1.89 ±1.47	-0.66±1.35	+0.69 ±0.65	+0.15 ±0.09	-0.03±0.17
+ frozen BERT	none	21.29 ±1.76	15.13 ±0.74	15.82 ±0.57	57.27 ±1.63	21.79 ±1.10
	on domain query	21.99 ±1.38	15.42 ±0.63	16.57 ±0.80	57.30 ±1.59	21.80 ±1.08
		+0.70 ±0.52	+0.29 ±0.43	+0.75 ±0.57	+0.04 ±0.06	+0.01 ±0.05
w/ triple attn.	none	21.55 ±1.55	16.64±0.42	17.10±1.26	52.96 ±3.66	21.98 ±1.38
	on domain attn.	25.14 ±4.03	**17.21** ±0.52	**18.11** ±1.56	53.15 ±3.60	22.09 ±1.41
		+3.59 ±3.20	+0.57 ±0.20	+1.02±0.50	+0.19±0.16	+0.12±0.11
+ frozen BERT	none	22.01 ±1.03	14.41 ±0.84	13.84 ±0.88	58.36 ±0.98	22.29 ±0.27
	on domain attn.	**26.56** ±0.78	15.93 ±0.98	15.20 ±1.19	58.53 ±0.97	**22.31** ±0.40
		+4.56 ±0.59	+1.52 ±0.90	+1.36 ±0.87	+0.17 ±0.04	+0.02 ±0.17

TABLEAU 6.12 – JGA de différentes variantes de SUMBT sur les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.1, avec et sans modulation de l'attention durant l'évaluation. Les informations supplémentaires données dans le tableau 6.9 s'appliquent ici aussi.

De manière surprenante, nous observons aussi que dans la plupart des cas, appliquer la modulation sur les attentions liées au nom, au type de valeurs, et au domaine ne baisse pas forcément la proportion de slots prédits avec la valeur `none` plus que dans le cas où la modulation est appliquée seulement à l'attention liée au domaine. Ceci semble nous montrer plusieurs choses : le modèle ne repose pas uniquement sur la/les couche(s) d'attention pour prédire une valeur ; les couches du reste du modèle ont une influence importante sur les prédictions, comme on le suppose le RNN via la considération des états cachés du tour de dialogue précédent ; et, la modulation de l'attention telle que proposée n'est pas adaptée pour être appliquée à autre chose que l'attention liée au domaine.

Pourquoi la modulation ne fonctionne pas sur les domaines `taxi` et `train` ?

Nous avons vu dans les tableaux 6.11 et 6.10 que la modulation ne semble pas fonctionner sur les domaines `taxi` et `train`. Cependant, grâce à des analyses supplémentaires sur MultiWOZ 2.0 nous pouvons observer que la modulation n'a pas le même effet sur les deux domaines. Pour le domaine `train`, les valeurs différentes de `none` semblent bénéficier de la modulation de manière similaire au domaine

Version	Modulation	Attraction	Hotel	Restaurant	Taxi	Train
train	-	73.0	69.7	71.1	73.3	71.8
dev	-	71.5	67.6	69.8	71.9	70.8
test	-	57.7	67.3	56.0	73.0	49.6
Original	none	99.1	83.1	79.4	96.0	84.0
	on slot attn.	97.6 (-1.5)	82.7 (-0.4)	78.0 (-1.4)	95.9 (-0.1)	84.9 (+0.9)
+ frozen BERT	none	99.0	89.2	87.2	96.4	94.3
	on slot attn.	92.6 (-6.4)	86.4 (-2.8)	77.9 (**-9.3**)	95.8 (-0.6)	92.8 (-1.5)
w/ triple queries	none	80.8	83.8	76.8	93.2	85.8
	on domain query	**78.6** (-2.2)	82.7 (-1.1)	**75.2** (-1.6)	**92.0** (-1.2)	85.4 (-0.4)
	on all queries	79.0 (-1.8)	82.6 (-1.2)	**75.2** (-1.6)	92.5 (-0.7)	85.2 (-0.6)
+ frozen BERT	none	83.3	80.6	81.8	94.9	93.1
	on domain query	81.1 (-2.2)	78.7 (-1.9)	79.0 (-2.8)	93.7 (-1.2)	90.8 (-2.3)
	on all queries	81.6 (-1.7)	**77.8** (-2.8)	78.1 (-3.7)	93.8 (-1.1)	90.0 (-3.1)
w/ triple attn.	none	93.2	81.5	80.0	95.5	81.6
	on domain attn.	87.7 (-5.5)	80.5 (-1.0)	76.6 (-3.4)	94.0 (**-1.5**)	**75.8** (**-5.8**)
	on all attn.	88.8 (-4.4)	80.3 (-1.2)	77.4 (-2.6)	94.3 (-1.2)	76.1 (-5.5)
+ frozen BERT	none	93.6	88.8	84.2	98.7	95.4
	on domain attn.	84.1 (**-9.5**)	83.7 (-5.1)	81.5 (-2.7)	97.6 (-1.1)	89.9 (-5.5)
	on all attn.	85.6 (-8.0)	83.1 (**-5.7**)	80.7 (-3.5)	97.7 (-1.0)	89.7 (-5.7)

TABLEAU 6.13 – Proportion (en pourcentage) de slots pour lesquels la valeur `none` est prédite, par domaine cible et par variante de SUMBT pour les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.0, avec et sans modulation de l'attention durant l'évaluation. Les 3 premières lignes correspondent à la proportion (en pourcentage) de slots pour lesquels la valeur `none` est la valeur de référence pour les jeux de données d'entraînement *train* et *dev*, et le jeu de données d'évaluation *test* de chaque domaine cible.

`restaurant`, avec par exemple une amélioration de l'accuracy pour les slots dont la valeur de référence est différente de `none` de respectivement +6,5 et +5,6 points pour la variante *avec triple attention* dont les poids de $BERT$ ont été fixés. Cependant, sur le domaine `train` la modulation de l'attention diminue aussi l'accuracy pour les slots dont la valeur de références est `none` (-0,6 contre -0,1 pour le domaine `restaurant`), ce qui fait que les deux effets se compensent et qu'aucune amélioration globale ne peut être notée sur `train`. Sur `taxi` au contraire, les valeurs différentes de `none` ne semblent pas bénéficier de la modulation de l'attention.

Néanmoins les observations sur le domaine `train` nous laissent penser que la métrique d'évaluation utilisée pour optimiser l'entraînement (la JGA) n'est pas la plus adaptée pour la tâche de DST et particulièrement pour le transfert zero-shot. En effet, les valeurs `none` semblent avoir plus d'impact sur la JGA que les valeurs différentes de `none`, ce qui peut pousser le modèle à davantage prédire la valeur

`none`.

Nous avons aussi cherché à comprendre si le faible impact de la modulation sur les domaines `taxi` et `train` pourrait s'expliquer par les types de slots à prédire dans ces deux domaines. Pour ce faire, la table 6.14 renseigne l'accuracy pour les slots dont la valeur de référence est différente de `none` par type de slot pour la variante *avec triple attention*. Nous pouvons observer que la modulation semble particulièrement bénéficier aux types de slots dont les noms sont commun avec d'autres domaines comme `area`, `leave at`, `book people`, `arrive by` et `name`. Cependant ce constat ne semble pas vraiment s'appliquer aux noms `departure` et `destination` tous deux communs aux domaines `taxi` et `train`. Il semble donc que l'on peut imputer le faible apport de la modulation sur ces deux domaines - pour la variante *avec triple attention* - à la présence de ces deux types de slots. Cependant, alors qu'ils sont présents dans ces deux domaines et qu'ils partagent des valeurs communes, la raison pour laquelle ces types de slots en particulier bénéficient si peu du transfert et de la modulation reste une question ouverte.

Limites des expériences sur SUMBT

Le modèle SUMBT est adapté aux systèmes de dialogue se reposant sur une base de données puisque l'ontologie peut être extraite directement de la base de données. De plus, en théorie il présente l'avantage de permettre de lier directement les critères mentionnés par l'utilisateur aux entités de la base de données. Cependant le modèle présente certaines limites. En effet, il n'est pas adapté à tous les types de données, comme par exemple les heures ou des nombres (personnes, étoiles, etc.). Le modèle n'est pas non plus adapté sur le long terme dans le cas où de plus en plus de valeurs et de types de slots seraient ajoutés à l'ontologie. En effet, nous supposons qu'il est plus difficile pour SUMBT de discriminer des valeurs d'un type de slot plus elles sont nombreuses. De plus, l'augmentation du nombre de valeurs et du nombre de types de slots augmente par la même occasion les calculs nécessaires. En effet, pour chaque type de slot les distances entre la sortie du modèle et toutes les représentation des valeurs associées au type de slot en entrée doivent être calculées.

D'un autre côté, si nous avons pu montrer que nos variantes et la modulation permettent d'améliorer le transfert zero-shot entre domaines de SUMBT, ces améliorations ne permettent pas d'atteindre les performances entre domaines zero-shot des modèles à l'état de l'art sur MultiWOZ 2.1 comme présenté dans le tableau 6.15. À noter que les variantes de SUMBT « w/ triple attn. » sont comparables en termes de nombre de paramètres uniquement au travail de [Li et al., 2021] lorsqu'ils utilisent GPT2 dans sa version basique.

slot type	MultiWOZ 2.0		MultiWOZ 2.1	
	not none acc.	**not none ref**	**not none acc.**	**not none ref**
hotel-parking	0.0 (+0.0)	28.5	0.2 (+0.0)	33.1
restaurant-food	0.0 (+0.0)	57.1	0.0 (+0.0)	60.7
hotel-internet	0.0 (+0.0)	29.1	0.0 (+0.0)	35.0
restaurant-book time	0.2 (+0.0)	37.1	2.5 (+0.7)	38.3
attraction-type	0.3 (+0.0)	49.3	0.3 (+0.1)	53.0
taxi-departure	3.3 (+0.0)	36.2	7.9 (+0.6)	36.6
hotel-type	1.2 (+0.1)	35.7	4.3 (+1.2)	40.2
hotel-stars	2.8 (+0.3)	33.0	3.4 (+0.3)	37.0
train-departure	3.5 (+0.7)	67.3	7.6 (+1.5)	68.1
taxi-destination	1.1 (+0.8)	36.5	2.5 (+0.1)	37.3
train-destination	6.5 (+1.4)	69.9	11.1 (+1.8)	70.7
hotel-price range	79.6 (+1.7)	39.4	86.5 (+1.0)	44.4
hotel-book people	42.9 (+1.9)	31.1	57.9 (+4.0)	32.1
hotel-name	12.4 (+2.1)	35.8	33.9 (+3.9)	43.6
hotel-book stay	23.3 (+2.7)	31.1	20.0 (+2.4)	32.6
hotel-book day	66.5 (+2.9)	31.5	89.2 (+3.6)	32.6
train-day	15.6 (+4.8)	68.3	12.0 (+3.6)	69.6
taxi-leave at	17.0 (+5.8)	20.4	19.8 (+6.9)	20.9
restaurant-name	13.5 (+5.9)	31.5	17.3 (+3.5)	47.3
restaurant-book people	52.3 (+6.7)	37.5	66.3 (+3.5)	38.9
restaurant-price range	78.8 (+7.5)	52.0	84.8 (+1.9)	57.4
restaurant-book day	57.7 (+7.8)	37.3	70.7 (+4.6)	38.3
attraction-name	13.8 (+8.7)	31.2	16.2 (+8.1)	43.1
hotel-area	55.9 (+10.1)	31.6	61.6 (+8.9)	39.2
train-arrive by	67.8 (+10.3)	38.8	56.7 (+8.7)	41.8
taxi-arrive by	25.9 (+11.2)	14.7	28.4 (+13.6)	14.8
train-book people	37.3 (+12.2)	29.2	39.2 (+9.4)	30.0
restaurant-area	61.3 (+18.5)	55.2	74.2 (+8.8)	59.4
train-leave at	56.4 (+19.5)	29.0	47.5 (+9.1)	32.7
attraction-area	43.3 (+27.0)	46.3	48.6 (+27.5)	52.7

TABLEAU 6.14 – Slot accuracy pour les valeurs différentes de `none` de la variante *avec triple attention* avec modulation moyennée sur 5 graines aléatoires, soit la proportion en pourcentage de valeurs différentes de `none` qui sont annotées correctement (« not none acc. »). Le gain apporté par la modulation est spécifié entre parenthèses et les types de slots sont ordonnés par gain croissant sur MultiWOZ 2.0. Pour se rendre compte de l'influence du type de slot est aussi reporté la proportion de valeurs différentes de `none` dans les références sur le nombre de valeurs à prédire dans le dataset de test (égale au nombre de tours de dialogue) par type de slot (« not none ref »).

Model	Attraction	Hotel	Restaurant	Taxi	Train
SUMBT *w/ triple attn.* + modulation	25.14	17.21	18.11	53.15	22.09
+ frozen BERT	26.56	15.93	15.20	58.53	22.31
[Li et al., 2021] (GPT2)	23.67	18.54	21.05	59.10	24.23
[Li et al., 2021] (GPT2-medium)	31.31	24.41	26.17	59.61	29.07
[Zhao et al., 2022]	56.4	21.8	38.2	78.4	38.7

TABLEAU 6.15 – JGA de plusieurs modèles pour les expériences de transfert zero-shot entre domaines sur MultiWOZ 2.1.

6.7 Conclusion

Dans ce chapitre, nous nous sommes d'abord intéressés aux capacités de généralisation et de transfert de SUMBT sur de nouvelles valeurs de slot. Nous avons montré de manière empirique que SUMBT n'est pas en mesure de généraliser sur des paires de slots qu'il n'a jamais vues et dont les valeurs n'ont jamais été rencontrées avec d'autres types de slots au cours de son entraînement. Cependant nous avons aussi montré que SUMBT était capable de transférer ses connaissances entre slots lorsque des valeurs de slots sont communes à plusieurs types de slots. Suite à ces observations, nous nous sommes demandé si le transfert pouvait aussi être observé dans le cadre de l'application de SUMBT à de nouveaux types de slots associés à de nouveaux domaines. Nous avons aussi proposé différentes variantes de SUMBT et introduit la *modulation de l'attention* dans le but d'améliorer le transfert. Nous avons constaté que cette méthode améliore avec succès les résultats originaux de SUMBT sur MultiWOZ 2.0 dans les domaines `attraction` et `hotel`, respectivement de 6,26 et 2,58 points avec la variante *avec triple attention*, sans nécessiter d'entraînement supplémentaire et sans jamais détériorer les résultats originaux.

Plusieurs expériences seraient intéressantes à réaliser afin de continuer les expérimentations sur ce sujet. Par exemple, une variable β pourrait être introduite à la place de la valeur 0 dans l'équation 6.5 pour étudier comment le changement de la valeur de β peut affecter les résultats de l'évaluation avec modulation. La possibilité d'étendre la modulation de l'attention à d'autres architectures serait aussi une direction de recherche intéressante.

Chapitre 7

Conclusions et perspectives

Dans ce mémoire, nous nous sommes intéressés aux systèmes de dialogue apprenant tout au long de leur vie, en nous concentrant sur les questions de recherche suivantes :

- *Comment définir un système de dialogue apprenant tout au long de sa vie ?*
- *Comment construire un tel système ? Quelles techniques et méthodologies peuvent être employées ?*
- *Comment évaluer un tel système ?*

L'apprentissage tout au long de sa vie, aussi nommé *Lifelong Learning* (LL), répond au défi majeur auquel beaucoup de systèmes sont confrontés en pratique. En effet, une fois déployé, un système va évoluer dans un environnement ouvert, où de nouveaux éléments peuvent apparaître au cours du temps. De plus, les exigences concernant le système peuvent elles aussi évoluer au cours du temps et ainsi introduire de nouveaux éléments. Le système devrait alors être en mesure de s'adapter à ces nouveaux éléments. L'application du LL aux systèmes de dialogue leur permettrait de s'adapter de manière efficace à l'évolution constante de leur environnement et des exigences de leurs concepteurs et de leurs utilisateurs. Dans ce mémoire nous nous concentrons particulièrement à l'amélioration de la compréhension de la langue, via l'étude des tâches de détection des slots et de suivi de l'état du dialogue.

7.1 Résumé et contributions

Nous résumons ici ce qui a été présenté dans les principaux chapitres de ce mémoire ainsi que les contributions principales.

Chapitre 3 : Le Lifelong Learning

Dans ce chapitre, nous définissons le LL comme la capacité d'un système à être appliqué à et à apprendre plusieurs tâches au cours du temps, en production, en autonomie, en continu et de manière interactive. L'apprentissage *multi-tâche au cours du temps* implique la nécessité du système à conserver de bonnes performances sur l'ensemble de ses tâches, en minimisant l'oubli et en maximisant le transfert de connaissances entre tâches. L'apprentissage *en production*, quant à lui, impose que l'apprentissage, ou du moins l'adaptation du système, se fasse après que le système ait été déployé, sur un flux de données non ordonnées pouvant présenter des éléments inconnus pour le système. L'apprentissage doit de plus se faire *en autonomie*, de telle sorte que l'apprentissage doit être pro-actif et que les données nécessaires à l'adaptation du système doivent être collectées et identifiées par le système lui-même, sans l'aide d'experts. L'adaptation doit aussi se faire *en continu*, de manière à ce que les nouveaux éléments soient immédiatement connus du système dès qu'ils ont été identifiés. Enfin, l'adaptation du système doit se faire *en interaction* avec son environnement. Trois étapes principales à réaliser par le système peuvent être identifiées : (1) Détecter la présence d'un nouvel élément, (2) Extraire et identifier le nouvel élément, et (3) Adapter les composants du système associés à ce nouvel élément. Appliqué à l'apprentissage supervisé, si les nouveaux éléments concernent uniquement de nouvelles instances et non de nouvelles classes ou de nouvelles tâches, alors les étapes sont les mêmes mais nous parlons plutôt d'*apprentissage sur le terrain*.

Par la suite, nous avons vu que de nombreux paradigmes sont liés en partie au LL. Ainsi, la première étape du LL est étudiée actuellement dans le cadre de l'apprentissage en monde ouvert, bien que beaucoup de domaines ne sont aujourd'hui pas couverts. La troisième étape a été étudiée plus largement dans le cadre de l'apprentissage continu et de l'apprentissage profond. De son côté, la notion de transfert a été étudiée dans le cadre de plusieurs paradigmes, comme l'apprentissage zero-shot, l'apprentissage sur peu d'exemples, l'apprentissage multi-tâche et le meta learning. L'apprentissage en interaction a été en partie étudié via l'apprentissage par renforcement et l'apprentissage actif. Enfin l'apprentissage en continu sur un flux de données non ordonnées a été étudié via l'apprentissage en ligne. Ainsi nous avons pu constater que le LL est complexe et qu'il est lié à plusieurs paradigmes d'apprentissage qui constituent individuellement encore aujourd'hui des sujets de recherche importants.

Appliqué aux systèmes de dialogue orientés tâche, le LL permettrait à ces systèmes de s'adapter efficacement à l'évolution constante de son environnement, par exemple dans le cas de la création d'un nouveau restaurant ou du déploiement du système dans une autre ville pour un système de dialogue pensé pour la re-

cherche et la réservation de restaurants. Le LL permettrait aussi à ces systèmes de s'adapter efficacement à l'évolution des exigences de ses concepteurs et de ses utilisateurs, par exemple dans le cas où de nouveaux critères de recherche peuvent être intégrés ou encore dans le cas où le système doit être déployé sur d'autres langues ou d'autres domaines. Un système de dialogue apprenant tout au long de sa vie pourra de plus profiter des interactions avec ses utilisateurs afin de réaliser les trois étapes du LL énoncés précédemment.

Résumé des contributions :

— Définition du LL.
— Explication des différences entre le LL et autres paradigmes connus.
— Application du LL aux systèmes de dialogue.

Chapitre 4 : Application et évaluation de l'apprentissage sur le terrain

Dans ce chapitre nous avons décrit une première version d'une méthodologie générale pour l'évaluation continue de l'apprentissage sur le terrain d'un système de dialogue. Ces travaux mettent de côté l'aspect multi-tâche du LL afin de permettre la comparaison avec d'autres travaux. En effet, il n'existe aujourd'hui pas à notre connaissance de système de dialogue capable d'apprendre tout au long de sa vie et l'apprentissage sur le terrain est encore peu étudié. Notre méthodologie d'évaluation est la première de ce type à être clairement définie.

Cette méthodologie consiste, de manière globale, à simuler les interactions avec les utilisateurs et à évaluer après chaque interaction les performances du système courant sur les jeux de données suivants : $test_{INITIAL}$, constitué de données similaires aux données initiales pour évaluer l'oubli ; $test_{LEARN}$, constitué de données contenant les éléments devant être appris au cours de la simulation ; et, $test_{UNKNOWN}$, constitué de données contenant des éléments n'apparaissant ni dans les données initiales, ni dans les données de simulation, pour évaluer la généralisation. Nous avons aussi construit un système de dialogue orienté tâche, qui peut améliorer de manière autonome et continue ses performances sur la tâche de détection des slots grâce à ses interactions avec ses utilisateurs.

Nous avons par la suite appliqué la méthodologie d'évaluation définie plus tôt à notre système via la simulation d'un utilisateur. Les expériences nous ont permis de voir les limitations d'une évaluation reposant sur des jeux de données de test générés à partir de patrons de phrases et de listes de mentions. Contrairement aux protocoles d'évaluation utilisés dans d'autres travaux, cette méthodologie nous a aussi permis de comparer, au cours du temps, les différentes méthodes d'adaptation et nous a permis d'identifier certaines de leurs différences.

Résumé des contributions :

- Une première définition d'une méthodologie d'évaluation générale pour les systèmes de dialogue apprenant sur le terrain.
- Un système de dialogue orienté tâche qui améliore sur le terrain sa détection des slots via la collection et l'inférence de nouveaux exemples d'entraînement grâce à ses interactions avec ses utilisateurs.
- L'évaluation du composant de compréhension de la langue du système précédent selon la méthodologie définie, avec simulation de l'utilisateur et simulation de la complétion de la base de connaissances associée au système.

Chapitre 5 : Étude du transfert entre langues dans le cadre de l'apprentissage continu

Dans ce chapitre, nous avons présenté une analyse du transfert entre langues dans le cadre de l'apprentissage continu pour la tâche d'étiquetage de séquences en utilisant le modèle multilingue BERT [Devlin et al., 2019b] ainsi que les corpus MultiATIS++ [Xu et al., 2020b] et MultiCoNER [Malmasi et al., 2022a]. En effet, la tâche de détection des slots, comprise dans le module de compréhension des systèmes de dialogue orientés tâche, peut être vue comme une tâche d'étiquetage de séquences. En plus d'être directement liée à l'étude du LL, cette manière d'étudier le transfert entre langues est à la fois novatrice et adaptée à un cas réel, ce qui explique son intérêt.

Notre principale conclusion suggère que, bien que l'oubli soit présent à la fin de l'apprentissage continu, le transfert entre langues est conservé sous la forme d'un transfert vers l'avant, ce qui permet au modèle d'avoir des capacités de récupération substantielles. De plus, nous montrons empiriquement que 1) un transfert vers l'avant élevé est lié au fait que les paramètres du modèle tendent progressivement vers une meilleure initialisation multilingue, et 2) que la plupart des connaissances des langues passées sont stockées dans l'encodeur de représentation des tokens (BERT) et non dans le classificateur spécifique à la tâche. Enfin, nous constatons également que les métriques actuelles pour évaluer le transfert dans le cadre de l'apprentissage continu doivent être adaptées si nous voulons mieux estimer la distribution du transfert sur l'axe d'adaptation.

Résumé des contributions :

- Étude du transfert entre langues dans le cadre de l'apprentissage continu de la tâche de détection des slots sur une séquence de langues.
- Étude des capacités de transfert de BERT multilingue [Devlin et al., 2019b].
- Étude des capacités d'un modèle continu :
  - Expériences sur l'initialisation multilingue.

— Expériences sur la récupération rapide.

Chapitre 6 : Étude du transfert entre domaines dans le cadre de l'apprentissage zero-shot

Dans ce chapitre, nous nous sommes d'abord intéressés au transfert entre domaines dans le cadre de l'apprentissage zero-shot de la tâche de suivi de l'état du dialogue sur de nouvelles données. Ces expériences ont été réalisées sur un modèle déjà existant, nommé SUMBT [Lee et al., 2019], et sur le corpus multi-domaines MultiWOZ [Budzianowski et al., 2018a, Eric et al., 2020].

Dans un premier temps, nous nous sommes intéressés aux capacités de généralisation et de transfert de SUMBT sur de nouvelles valeurs de slot. Nous avons montré de manière empirique que SUMBT n'est pas en mesure de généraliser sur des paires de slots qu'il n'a jamais vues et dont les valeurs n'ont jamais été rencontrées avec d'autres types de slots au cours de son entraînement. Cependant nous avons aussi montré que SUMBT est capable de transférer ses connaissances entre slots, lorsque des valeurs sont communes à plusieurs types de slots.

Suite à ces observations, nous nous sommes demandé si le transfert pouvait aussi être observé dans le cadre de l'application de SUMBT à de nouveaux types de slots associés à de nouveaux domaines. Nous avons aussi proposé différentes variantes de SUMBT et introduit la *modulation de l'attention* dans le but d'améliorer le transfert. Nous avons constaté que cette méthode améliore avec succès les résultats originaux de SUMBT sur MultiWOZ 2.0 dans les domaines `attraction` et `hotel`, respectivement de 6,26 et 2,58 points avec la variante *avec triple attention*, sans nécessiter d'entraînement supplémentaire et sans jamais détériorer les résultats originaux.

Résumé des contributions :

— Étude des capacités de généralisation de SUMBT [Lee et al., 2019] sur la tâche de suivi de l'état du dialogue dans le cadre de nouvelles valeurs de slot.
— Étude des capacités de transfert de SUMBT entre types de slots partageant des valeurs communes.
— Proposition de plusieurs variantes de SUMBT.
— Proposition d'une méthode applicable pendant l'inférence appelée *modulation de l'attention*.
— Étude du transfert entre domaines dans le cadre de l'apprentissage zero-shot de nouveaux domaines et de la configuration « leave-one-out » sur MultiWOZ et avec les variantes et la modulation de l'attention décrites au préalable.

7.2 Perspectives

Suite aux travaux réalisés pour l'application et l'évaluation de l'apprentissage sur le terrain (chapitre 4), il serait intéressant d'adapter le système de dialogue à l'apprentissage sur le terrain de nouveaux slots et de nouveau domaines, afin de se rapprocher d'un contexte expérimental couvrant l'ensemble des aspect du LL. De plus, la méthodologie d'évaluation peut être combinée aux méthodologies d'évaluation décrites dans le cadre de l'apprentissage continu en ligne [Aljundi et al., 2019]. Il serait aussi intéressant d'ajouter à la méthodologie d'évaluation un moyen d'évaluer la robustesse de l'apprentissage sur le terrain du système face au bruit, par exemple si l'utilisateur fournit de mauvaises indications au système. Ceci permettrait de considérer la notion de fiabilité des nouveaux éléments sur lesquels le système s'adapte.

D'un autre côté, les études de transfert réalisées dans les chapitres 5 et 6 nous ont permis d'observer notamment les capacités de transfert du modèle BERT [Devlin et al., 2019b]. Ces observations rejoignent celles réalisées sur d'autres modèles reposant sur l'architecture `transformer` et montrent ainsi l'intérêt pour le LL de ces modèles pré-entraînés de manière auto-supervisée sur un large corpus de données. Cependant, bien que le développement de ces modèles soit encourageant pour le LL, nous avons pu voir que des travaux sont encore nécessaires afin de mieux comprendre les mécanismes de transfert mis en place par ces modèles.

En particulier, les conclusions et les observations des travaux sur l'étude du transfert dans le cadre continu nous ont montré plusieurs choses. Tout d'abord, il est important de diversifier les cadres d'études afin d'avoir une vision plus complète des mécanismes de transfert. Ces expériences nous ont aussi montré que des travaux sont encore nécessaires pour analyser et comprendre comment le transfert en avant et le transfert en arrière se comportent et aussi pour développer des méthodes visant à contrer le phénomène d'oubli de manière efficace. Elles mettent aussi en lumière de nouvelles directions de recherche, comme la combinaison de l'apprentissage continu et de l'apprentissage actif qui permettrait de se rapprocher du cadre du LL en offrant au système plus d'autonomie.

En parallèle, les expériences sur le transfert entre domaines dans le cadre de l'apprentissage zero-shot nous ont permis de voir qu'il serait intéressant de continuer à développer des méthodes visant à augmenter le transfert pendant l'inférence, à l'image de la modulation de l'attention que nous avons proposée. En effet, ces méthodes ne nécessitent aucun entraînement supplémentaire, ce qui est une caractéristique recherchée aujourd'hui. La modulation de l'attention pourra elle-même être étudiée plus en profondeur, en jouant sur certaines variables et en essayant de l'appliquer à d'autres cas de figure et d'autres architectures.

De manière générale, nous avons pu constater qu'il est capital de considérer le LL comme un objectif, puisqu'il permet de mettre en lumière de nouvelles directions de recherche, ainsi que le manque de travaux sur certains aspects du LL ou dans certains contextes expérimentaux. Les travaux que nous avons réalisés, bien que s'attaquant à ces aspects peu étudiés, ne permettent pas de résoudre l'ensemble des problèmes liés à la construction de systèmes de dialogue capables d'apprendre sur le terrain. Il est donc nécessaire, dans un premier temps que des travaux sur les paradigmes associées au LL, comme l'apprentissage zero-shot et l'apprentissage continu, continuent d'être réalisés ; et dans un second temps que des travaux combinant plusieurs paradigmes soient proposés, comme l'apprentissage actif et l'apprentissage continu ou bien l'apprentissage en ligne et l'apprentissage continu, tout en gardant en tête l'objectif du LL.

Bibliographie

[Abdi, 2007] Abdi, H. (2007). The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, pages 508–510.

[Albert and Ruiter, 2018] Albert, S. and Ruiter, J. D. (2018). Repair : The interface between interaction and cognition. *Topics in Cognitive Science*, 10 :279 – 313.

[Aljundi et al., 2018] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses : Learning what (not) to forget. In *ECCV*.

[Aljundi et al., 2019] Aljundi, R., Kelchtermans, K., and Tuytelaars, T. (2019). Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263.

[Arkhipov et al., 2019] Arkhipov, M., Trofimova, M., Kuratov, Y., and Sorokin, A. (2019). Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93.

[Arora et al., 2020] Arora, G., Jain, C., Chaturvedi, M., and Modi, K. (2020). HINT3 : Raising the bar for intent detection in the wild. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 100–105, Online. Association for Computational Linguistics.

[Arora et al., 2019] Arora, G., Rahimi, A., and Baldwin, T. (2019). Does an LSTM forget more than a CNN ? an empirical study of catastrophic forgetting in NLP. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, Sydney, Australia. Australasian Language Technology Association.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date : 07-05-2015 Through 09-05-2015.

[Béchet and Raymond, 2018] Béchet, F. and Raymond, C. (2018). Is ATIS too shallow to go deeper for benchmarking spoken language understanding models ? In *Proc. Interspeech 2018*, pages 3449–3453.

[Bender and Koller, 2020] Bender, E. M. and Koller, A. (2020). Climbing towards NLU : On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language modelgraves-2013-bilstm. *J. Mach. Learn. Res.*, 3(null) :1137–1155.

[Bengio et al., 2009] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *ICML '09*.

[Bobrow et al., 1977] Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). Gus, a frame-driven dialog system. *Artificial Intelligence*, 8(2) :155–173.

[Bordes et al., 2017] Bordes, A., Boureau, Y.-L., and Weston, J. (2017). Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

[Brabra et al., 2022] Brabra, H., Baez, M., Benatallah, B., Gaaloul, W., Bouguelia, S., and Zamanirad, S. (2022). Dialogue management in conversational systems : a review of approaches, challenges, and opportunities. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–15.

[Brown et al., 2020a] 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., and Amodei, D. (2020a). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

[Brown et al., 2020b] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020b). Language models are few-shot learners. *Advances in neural information processing systems*, 33 :1877–1901.

[Budzianowski et al., 2018a] Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018a). MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

[Budzianowski et al., 2018b] Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018b). MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

[Campagna et al., 2020] Campagna, G., Foryciarz, A., Moradshahi, M., and Lam, M. (2020). Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

[Carlson et al., 2010] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E., and Mitchell, T. (2010). Toward an architecture for never-ending language learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1) :1306–1313.

[Casanueva et al., 2020] Casanueva, I., Temcinas, T., Gerz, D., Henderson, M., and Vulic, I. (2020). Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at https ://github.com/PolyAI-LDN/task-specific-datasets.

[Cattan et al., 2021] Cattan, O., Rosset, S., and Servan, C. (2021). On the cross-lingual transferability of multilingual prototypical models across NLU tasks. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, pages 36–43, Online. Association for Computational Linguistics.

[Chen et al., 2019] Chen, Q., Zhuo, Z., and Wang, W. (2019). BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

[Chen and Liu, 2016] Chen, Z. and Liu, B. (2016). *Lifelong Machine Learning*. Morgan & Claypool Publishers.

[Chen et al., 2018] Chen, Z., Liu, B., Brachman, R., Stone, P., and Rossi, F. (2018). *Lifelong Machine Learning : Second Edition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

[Clark et al., 2019] Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What does BERT look at ? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

[Conneau et al., 2020a] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020a). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

[Conneau et al., 2020b] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020b). Unsupervised cross-lingual representation learning at scale. In *ACL*.

[Coria et al., 2021] Coria, J. M., Bredin, H., Ghannay, S., and Rosset, S. (2021). Overlap-aware low-latency online speaker diarization based on end-to-end local segmentation. *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1139–1146.

[Coria et al., 2022] Coria, J. M., Veron, M., Ghannay, S., Bernard, G., Bredin, H., Galibert, O., and Rosset, S. (2022). Analyzing BERT cross-lingual transfer capabilities in continual sequence labeling. In *Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models*, pages 15–25, Virtual. International Conference on Computational Linguistics.

[Coucke et al., 2018] Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., Primet, M., and Dureau, J. (2018). Snips voice platform : an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

[Crawshaw, 2020] Crawshaw, M. (2020). Multi-task learning with deep neural networks : A survey. *CoRR*, abs/2009.09796.

[Cuayáhuitl, 2017] Cuayáhuitl, H. (2017). *SimpleDS : A Simple Deep Reinforcement Learning Dialogue System*, pages 109–118. Springer Singapore, Singapore.

[Dai et al., 2021] Dai, Y., Li, H., Li, Y., Sun, J., Huang, F., Si, L., and Zhu, X. (2021). Preview, attend and review : Schema-aware curriculum learning for multi-domain dialogue state tracking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2 : Short Papers)*, pages 879–885, Online. Association for Computational Linguistics.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[Deriu et al., 2020] Deriu, J., Rodrigo, A., Otegi, A., Echegoyen, G., Rosset, S., Agirre, E., and Cieliebak, M. (2020). Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, pages 1–56.

[Deshpande et al., 2022] Deshpande, A., Talukdar, P., and Narasimhan, K. (2022). When is BERT multilingual ? isolating crucial ingredients for cross-lingual transfer. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 3610–3623, Seattle, United States. Association for Computational Linguistics.

[Devlin et al., 2019a] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019a). BERT : Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[Devlin et al., 2019b] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019b). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[Do et al., 2020] Do, Q., Gaspers, J., Roeding, T., and Bradford, M. (2020). To what degree can language borders be blurred in BERT-based multilingual spoken language understanding ? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2699–2709, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[Dyer et al., 2015] Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2) :179–211.

[Eric et al., 2020] Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., Kumar, A., Goyal, A., Ku, P., and Hakkani-Tur, D. (2020). MultiWOZ 2.1 : A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation*

Conference, pages 422–428, Marseille, France. European Language Resources Association.

[FitzGerald et al., 2022] FitzGerald, J., Hench, C., Peris, C., Mackie, S., Rottmann, K., Sanchez, A., Nash, A., Urbach, L., Kakarala, V., Singh, R., Ranganath, S., Crist, L., Britan, M., Leeuwis, W., Tur, G., and Natarajan, P. (2022). Massive : A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages.

[French, 1999] French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4) :128 – 135.

[Ghannay et al., 2020] Ghannay, S., Neuraz, A., and Rosset, S. (2020). What is best for spoken language understanding : small but task-dependant embeddings or huge but out-of-domain embeddings ? In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8114–8118.

[Graves et al., 2013] Graves, A., Jaitly, N., and Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

[Gupta et al., 2022] Gupta, R., Lee, H., Zhao, J., Cao, Y., Rastogi, A., and Wu, Y. (2022). Show, don't tell : Demonstrations outperform descriptions for schema-guided task-oriented dialogue. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 4541–4549, Seattle, United States. Association for Computational Linguistics.

[Hadsell et al., 2020] Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing Change : Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 24 :1028–1040.

[Ham et al., 2020] Ham, D., Lee, J.-G., Jang, Y., and Kim, K.-E. (2020). End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. Association for Computational Linguistics.

[Hancock et al., 2019] Hancock, B., Bordes, A., Mazare, P.-E., and Weston, J. (2019). Learning from dialogue after deployment : Feed yourself, chatbot ! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

[Heck et al., 2020] Heck, M., van Niekerk, C., Lubis, N., Geishauser, C., Lin, H.-C., Moresi, M., and Gasic, M. (2020). TripPy : A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual*

Meeting of the Special Interest Group on Discourse and Dialogue, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

[Heinzerling, 2019] Heinzerling, B. (2019). Nlp's clever hans moment has arrived. *The Gradient*.

[Hemphill et al., 1990a] Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990a). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language : Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

[Hemphill et al., 1990b] Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990b). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language : Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

[Hendrycks and Gimpel, 2017] Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8) :1735–1780.

[Hough, 2014] Hough, J. (2014). *Modelling Incremental Self-Repair Processing in Dialogue.* PhD thesis, Queen Mary University of London.

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

[Jacqmin, 2022] Jacqmin, L. (2022). " Est-ce que tu me suis ? " : une revue du suivi de l'état du dialogue. In Estève, Yannick, Jiménez, Tania, Parcollet, Titouan, Boito, Z., and Marcely, editors, *Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 2 : 24e Rencontres Etudiants Chercheurs en Informatique pour le TAL (RECITAL)*, pages 1–19, Avignon, France. ATALA.

[Joulin et al., 2017] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

[K et al., 2020] K, K., Wang, Z., Mayhew, S., and Roth, D. (2020). Cross-Lingual Ability of Multilingual BERT : An Empirical Study. In *International Conference on Learning Representations*.

[Kalchbrenner et al., 2014] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

[Kale and Rastogi, 2020] Kale, M. and Rastogi, A. (2020). Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

[Ke et al., 2021] Ke, Z., Xu, H., and Liu, B. (2021). Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 4746–4755, Online. Association for Computational Linguistics.

[Klie et al., 2021] Klie, J.-C., Eckart de Castilho, R., and Gurevych, I. (2021). Human-in-the-LoopEntity linking for low resource domains. In *Proceedings of the Second Workshop on Data Science with Human in the Loop : Language Advances*, pages 41–43, Online. Association for Computational Linguistics.

[Komeili et al., 2022] Komeili, M., Shuster, K., and Weston, J. (2022). Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 8460–8478, Dublin, Ireland. Association for Computational Linguistics.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[Langkilde and Knight, 1998] Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada. Association for Computational Linguistics.

[Larson et al., 2019] Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., and Mars, J. (2019). An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

[Lee et al., 2021a] Lee, H., Jo, S., Kim, H., Jung, S., and Kim, T.-Y. (2021a). Sumbt+larl : Effective multi-domain end-to-end neural task-oriented dialog system. *IEEE Access*, 9 :116133–116146.

[Lee et al., 2019] Lee, H., Lee, J., and Kim, T.-Y. (2019). SUMBT : Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

[Lee et al., 2022] Lee, H.-y., Li, S.-W., and Vu, T. (2022). Meta learning for natural language processing : A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 666–684, Seattle, United States. Association for Computational Linguistics.

[Lee, 2017] Lee, S. (2017). Toward continual learning for conversational agents. *arXiv preprint arXiv :1712.09943*.

[Lee et al., 2021b] Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. (2021b). Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *5th Annual Conference on Robot Learning*.

[Lee, 2021] Lee, Y. (2021). Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics : EMNLP 2021*, pages 1296–1303, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Lesort et al., 2019] Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. (2019). Continual Learning for Robotics : Definition, Framework, Learning Strategies, Opportunities and Challenges. *Information Fusion*.

[Lewis et al., 2020] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART : Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

[Li et al., 2017] Li, J., Miller, A. H., Chopra, S., Ranzato, M., and Weston, J. (2017). Learning through dialogue interactions by asking questions. In *International Conference on Learning Representations*.

[Li et al., 2021] Li, S., Cao, J., Sridhar, M., Zhu, H., Li, S.-W., Hamza, W., and McAuley, J. (2021). Zero-shot generalization in dialog state tracking through generative question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics : Main Volume*, pages 1063–1074, Online. Association for Computational Linguistics.

[Lin et al., 2021a] Lin, Z., Liu, B., Madotto, A., Moon, S., Zhou, Z., Crook, P., Wang, Z., Yu, Z., Cho, E., Subba, R., and Fung, P. (2021a). Zero-shot dialogue state tracking via cross-task transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7890–7900, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Lin et al., 2021b] Lin, Z., Liu, B., Moon, S., Crook, P., Zhou, Z., Wang, Z., Yu, Z., Madotto, A., Cho, E., and Subba, R. (2021b). Leveraging slot descriptions for zero-shot cross-domain dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 5640–5648, Online. Association for Computational Linguistics.

[Liu, 2020] Liu, B. (2020). Learning on the job : Online lifelong and continual learning. In *Proceedings of 34th AAAI Conference on Artifical Intelligence (AAAI-2020)*, New York City, USA.

[Liu and Lane, 2016] Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*, pages 685–689.

[Liu et al., 2021a] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021a). Pre-train, prompt, and predict : A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

[Liu et al., 2021b] Liu, Z., Winata, G. I., Madotto, A., and Fung, P. (2021b). Preserving Cross-Linguality of Pre-trained Models via Continual Learning. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 64–71, Online. Association for Computational Linguistics.

[Lopez-Paz and Ranzato, 2017] Lopez-Paz, D. and Ranzato, M. A. (2017). Gradient Episodic Memory for Continual Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Loshchilov and Hutter, 2019] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[Madotto et al., 2021] Madotto, A., Lin, Z., Zhou, Z., Moon, S., Crook, P., Liu, B., Yu, Z., Cho, E., Fung, P., and Wang, Z. (2021). Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Madotto et al., 2020] Madotto, A., Lin, Z., Zhou, Z., Moon, S., Crook, P., Liu, B., Yu, Z., Cho, E., and Wang, Z. (2020). Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv :2012.15504*.

[Madotto et al., 2018] Madotto, A., Wu, C.-S., and Fung, P. (2018). Mem2Seq : Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1468–1478, Melbourne, Australia. Association for Computational Linguistics.

[Malmasi et al., 2022a] Malmasi, S., Fang, A., Fetahu, B., Kar, S., and Rokhlenko, O. (2022a). MultiCoNER : a Large-scale Multilingual dataset for Complex Named Entity Recognition.

[Malmasi et al., 2022b] Malmasi, S., Fang, A., Fetahu, B., Kar, S., and Rokhlenko, O. (2022b). SemEval-2022 Task 11 : Multilingual Complex Named Entity Recognition (MultiCoNER). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

[Mazumder et al., 2019a] Mazumder, S., Liu, B., Wang, S., and Esmaeilpour, S. (2019a). Building an application independent natural language interface. *ArXiv*, abs/1910.14084.

[Mazumder et al., 2019b] Mazumder, S., Liu, B., Wang, S., and Ma, N. (2019b). Lifelong and interactive learning of factual knowledge in dialogues. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 21–31, Stockholm, Sweden. Association for Computational Linguistics.

[McCann et al., 2018] McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon : Multitask learning as question answering. *CoRR*, abs/1806.08730.

[McCloskey and Cohen, 1989] McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks : The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press.

[Mezza et al., 2018] Mezza, S., Cervone, A., Stepanov, E., Tortoreto, G., and Riccardi, G. (2018). ISO-standard domain-independent dialogue act tagging for conversational agents. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3539–3551, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space in 1st international conference on learning representations, iclr 2013, scottsdale, arizona, usa, may 2-4, 2013. In *Workshop Track Proceedings*.

[Mitchell et al., 2018] Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2018). Never-ending learning. *Commun. ACM*, 61(5) :103–115.

[Monaikul et al., 2021] Monaikul, N., Castellucci, G., Filice, S., and Rokhlenko, O. (2021). Continual learning for named entity recognition. In *AAAI*.

[Mueller et al., 2020] Mueller, D., Andrews, N., and Dredze, M. (2020). Sources of transfer in multilingual named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8093–8104, Online. Association for Computational Linguistics.

[Nair et al., 2021] Nair, A., Dalal, M., Gupta, A., and Levine, S. (2021). {AWAC} : Accelerating online reinforcement learning with offline datasets.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*.

[Nakayama, 2018] Nakayama, H. (2018). seqeval : A python framework for sequence labeling evaluation. Software available from https ://github.com/chakki-works/seqeval.

[Ni et al., 2022] Ni, J., Young, T., Pandelea, V., Xue, F., and Cambria, E. (2022). Recent advances in deep learning based dialogue systems : a systematic survey. *Artificial Intelligence Review*.

[Nilsback and Zisserman, 2008] Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729.

[Niven and Kao, 2019] Niven, T. and Kao, H.-Y. (2019). Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu : A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

[Parisi et al., 2019] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks : A review. *Neural Networks*, 113 :54–71.

[Peng et al., 2017] Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017). Composite task-completion dialogue policy learning via

hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240, Copenhagen, Denmark. Association for Computational Linguistics.

[Peng et al., 2020] Peng, B., Zhu, C., Li, C., Li, X., Li, J., Zeng, M., and Gao, J. (2020). Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics : EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). GloVe : Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Pires et al., 2019] Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual BERT ? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

[Qian et al., 2021] Qian, K., Beirami, A., Lin, Z., De, A., Geramifard, A., Yu, Z., and Sankar, C. (2021). Annotation inconsistency and entity bias in MultiWOZ. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 326–337, Singapore and Online. Association for Computational Linguistics.

[Qin et al., 2019] Qin, L., Che, W., Li, Y., Wen, H., and Liu, T. (2019). A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2078–2087, Hong Kong, China. Association for Computational Linguistics.

[Qin et al., 2021] Qin, L., Xie, T., Che, W., and Liu, T. (2021). A survey on spoken language understanding : Recent advances and new frontiers. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4577–4584. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

[Qureshi et al., 2020] Qureshi, A. H., Miao, Y., and Yip, M. C. (2020). Active continual learning for planning and navigation. In *ICML 2020 Workshop on Real World Experiment Design and Active Learning*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer

learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140) :1–67.

[Rahimi et al., 2019] Rahimi, A., Li, Y., and Cohn, T. (2019). Massively Multilingual Transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

[Rastogi et al., 2020] Rastogi, A., Zang, X., Sunkara, S., Gupta, R., and Khaitan, P. (2020). Towards scalable multi-domain conversational agents : The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

[Robins, 1995] Robins, A. (1995). Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2) :123–146.

[Roller et al., 2021] Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Smith, E. M., Boureau, Y.-L., and Weston, J. (2021). Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics : Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

[Ruder, 2017] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

[Ruder, 2018] Ruder, S. (2018). A Review of the Neural History of Natural Language Processing. http://ruder.io/a-review-of-the-recent-history-of-nlp/.

[Schuster et al., 2019] Schuster, S., Gupta, S., Shah, R., and Lewis, M. (2019). Cross-lingual transfer learning for multilingual task oriented dialog. In *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)*, pages 3795–3805, Minneapolis, Minnesota. Association for Computational Linguistics.

[Shah et al., 2019] Shah, D., Gupta, R., Fayazi, A., and Hakkani-Tur, D. (2019). Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490, Florence, Italy. Association for Computational Linguistics.

[Shuster et al., 2022] Shuster, K., Xu, J., Komeili, M., Ju, D., Smith, E. M., Roller, S., Ung, M., Chen, M., Arora, K., Lane, J., Behrooz, M., Ngan, W., Poff, S., Goyal, N., Szlam, A. D., Boureau, Y.-L., Kambadur, M., and Weston, J. (2022). Blenderbot 3 : a deployed conversational agent that continually learns to responsibly engage. *ArXiv*, abs/2208.03188.

[Socher et al., 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[Su et al., 2018] Su, S.-Y., Yuan, P.-C., and Chen, Y.-N. (2018). How time matters : Learning time-decay attention for contextual spoken language understanding in dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 2133–2142, New Orleans, Louisiana. Association for Computational Linguistics.

[Sun et al., 2020] Sun, F.-K., Ho, C.-H., and yi Lee, H. (2020). Lamol : Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

[Thrun, 1995] Thrun, S. (1995). Is learning the n-th thing any easier than learning the first ? In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.

[Thrun and Mitchell, 1995] Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1) :25–46. The Biology and Technology of Intelligent Autonomous Agents.

[Tjong Kim Sang, 2002] Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task : Language-independent named entity recognition. In *COLING-02 : The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

[Tjong Kim Sang and Buchholz, 2000] Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

[Todorov et al., 2012] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco : A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

[Tran and Nguyen, 2017] Tran, V.-K. and Nguyen, L.-M. (2017). Natural language generation for spoken dialogue system using RNN encoder-decoder networks. In *Proceedings of the 21st Conference on Computational Natural Language Lear-*

ning (CoNLL 2017), pages 442–451, Vancouver, Canada. Association for Computational Linguistics.

[Upadhyay et al., 2018] Upadhyay, S., Faruqui, M., Tür, G., Dilek, H.-T., and Heck, L. (2018). (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Vedula et al., 2022] Vedula, N., Gupta, R., Alok, A., Sridhar, M., and Ananthakrishnan, S. (2022). Advin : Automatically discovering novel domains and intents from user text utterances. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7627–7631.

[Veron, 2019] Veron, M. (2019). Lifelong learning et systèmes de dialogue : définition et perspectives (lifelong learning and dialogue system : definition and discussion). In *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles (TALN) PFIA 2019. Volume III : RECITAL*, pages 563–576, Toulouse, France. ATALA.

[Veron et al., 2022] Veron, M., Galibert, O., Bernard, G., and Rosset, S. (2022). Attention modulation for zero-shot cross-domain dialogue state tracking. In *Proceedings of the 3rd Workshop on Computational Approaches to Discourse*, pages 86–91, Gyeongju, Republic of Korea and Online. International Conference on Computational Linguistics.

[Veron et al., 2019] Veron, M., Ghannay, S., Ligozat, A.-L., and Rosset, S. (2019). Lifelong learning and task-oriented dialogue system : what does it mean ? In *International Workshop on Spoken Dialogue Systems Technology*, Siracusa, Italy.

[Veron et al., 2020] Veron, M., Rosset, S., Galibert, O., and Bernard, G. (2020). Evaluate on-the-job learning dialogue systems and a case study for natural language understanding. In *Workshop NeurIPS 2020 Human in the Loop Dialogue Systems*, Virtual only, United States.

[Wang et al., 2022] Wang, X., Chen, Y., and Zhu, W. (2022). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9) :4555–4576.

[Wang et al., 2020] Wang, Z., K, K., Mayhew, S., and Roth, D. (2020). Extending multilingual BERT to low-resource languages. In *Findings of the Association*

for Computational Linguistics : EMNLP 2020, pages 2649–2656, Online. Association for Computational Linguistics.

[Weston et al., 2015] Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. Publisher Copyright : © 2015 International Conference on Learning Representations, ICLR. All rights reserved. ; 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date : 07-05-2015 Through 09-05-2015.

[Wolf et al., 2017] Wolf, M., Miller, K., and Grodzinsky, F. (2017). Why we should have seen that coming : Comments on microsoft's tay "experiment," and wider implications. *The ORBIT Journal*, 1(2) :1–12.

[Wu et al., 2019] Wu, C.-S., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., and Fung, P. (2019). Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. Association for Computational Linguistics.

[Xia et al., 2022] Xia, Y., Wang, Q., Lyu, Y., Zhu, Y., Wu, W., Li, S., and Dai, D. (2022). Learn and review : Enhancing continual named entity recognition via reviewing synthetic samples. In *Findings of the Association for Computational Linguistics : ACL 2022*, pages 2291–2300, Dublin, Ireland. Association for Computational Linguistics.

[Xu et al., 2022] Xu, J., Ung, M., Komeili, M., Arora, K., Boureau, Y.-L., and Weston, J. (2022). Learning new skills after deployment : Improving open-domain internet-driven dialogue with human feedback. *ArXiv*, abs/2208.03270.

[Xu et al., 2020a] Xu, W., Haider, B., and Mansour, S. (2020a). End-to-end slot alignment and recognition for cross-lingual NLU. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063, Online. Association for Computational Linguistics.

[Xu et al., 2020b] Xu, W., Haider, B., and Mansour, S. (2020b). End-to-end slot alignment and recognition for cross-lingual NLU. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063, Online. Association for Computational Linguistics.

[Xue et al., 2020] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2020). mt5 : A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv :2010.11934*.

[Xue et al., 2021] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mT5 : A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics :*

Human Language Technologies, pages 483–498, Online. Association for Computational Linguistics.

[Young, 2006] Young, S. (2006). Using pomdps for dialog management. In *2006 IEEE Spoken Language Technology Workshop*, pages 8–13.

[Yu et al., 2004] Yu, C., Aoki, P. M., and Woodruff, A. (2004). Detecting user engagement in everyday conversations. *8th International Conference on Spoken Language Processing (ICSLP)*.

[Zang et al., 2020] Zang, X., Rastogi, A., Sunkara, S., Gupta, R., Zhang, J., and Chen, J. (2020). Multiwoz 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, pages 109–117.

[Zhang et al., 2021] Zhang, H., Xu, H., Lin, T.-E., and Lyu, R. (2021). Discovering new intents with deep aligned clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16) :14365–14373.

[Zhang et al., 2022] Zhang, J., Hashimoto, K., Wan, Y., Liu, Z., Liu, Y., Xiong, C., and Yu, P. (2022). Are pre-trained transformers robust in intent classification ? a missing ingredient in evaluation of out-of-scope intent detection. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 12–20, Dublin, Ireland. Association for Computational Linguistics.

[Zhang et al., 2019a] Zhang, Z., Huang, M., Zhao, Z., Ji, F., Chen, H., and Zhu, X. (2019a). Memory-augmented dialogue management for task-oriented dialogue systems. *ACM Trans. Inf. Syst.*, 37(3).

[Zhang et al., 2019b] Zhang, Z., Zhang, Z., Chen, H., and Zhang, Z. (2019b). A joint learning framework with bert for spoken language understanding. *IEEE Access*, 7 :168849–168858.

[Zhao et al., 2022] Zhao, J., Gupta, R., Cao, Y., Yu, D., Wang, M., Lee, H., Rastogi, A., Shafran, I., and Wu, Y. (2022). Description-driven task-oriented dialog modeling. *ArXiv*, abs/2201.08904.

[Zhao et al., 2021] Zhao, Y., Wang, Z., Zhu, C., and Wang, S. (2021). Efficient dialogue complementary policy learning via deep Q-network policy and episodic memory policy. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4311–4323, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Zhou et al., 2021] Zhou, Z., Shin, J. Y., Gurudu, S., Gotway, M., and Liang, J. (2021). Active, continual fine tuning of convolutional neural networks for reducing annotation efforts. *Medical image analysis*, 71 :101997.

[Zhu et al., 2019] Zhu, C., Zeng, M., and Huang, X. (2019). Multi-task learning for natural language generation in task-oriented dialogue. In *Proceedings of the*

2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1261–1266, Hong Kong, China. Association for Computational Linguistics.

Glossaire

token Un token est une unité de texte issu du processus tokenisation. Un token peut correspondre à un mot ou à un sous mot. La tokenisation consiste à segmenter un texte en token.

tokenisation La tokenisation consiste à segmenter un texte en tokens. La sortie de la tokenisation dépend de l'outil utilisé, appelé *tokenizer*. Par exemple, le modèle BERT [Devlin et al., 2019b] dispose de son propre tokenizer accessible via Huggingface, tel que la phrase `"I have a new GPU!"` resulte en la liste de tokens `["i", "have", "a", "new", "gp", "##u", "!"]`, les caractères `"##"` désignant la suite d'un mot.

tour de parole Dans le cadre d'une vision simplifiée d'un dialogue entre un utilisateur et un système, on suppose une alternance entre un énoncé utilisateur et un énoncé système. Dans ce cas, un tour de dialogue correspond à un énoncé système suivi d'un énoncé utilisateur (voir figure 6.1).

énoncé Unité de discours pouvant être composée d'un mot ou de plusieurs mais ne consistant pas nécessairement en une phrase complète. Pour les systèmes de dialogue orientés tâche on parle d'énoncé utilisateur ou système pour décrire ce que chacun dit/écrit. Dans le cas d'un système de dialogue orienté tâche pour la recherche de recettes de cuisine, on peut imaginer l'énoncé utilisateur suivant : `"Des suggestions pour un anniversaire ?"`.

Acronymes

DB Base de données, abréviation de l'anglais Data Base.

DST Suivi de l'état du dialogue, abréviation de l'anglais Dialogue State Tracking.

JGA Joint Goal Accuracy, mesure utilisée pour la tâche de DST définie comme le pourcentage de tours où l'ensemble des slots ont été correctement prédits..

KB Base de connaissances, abréviation de l'anglais Knowledge Base.

NER Reconnaissance des entités nommées, abréviation de l'anglais Named Entity Recognition.

NLG Génération de la langue, abréviation de l'anglais Natural Language Generation.

NLU Compréhension de la langue, abréviation de l'anglais Natural Language Understanding.

RL Apprentissage par renforcement, abréviation de l'anglais Reinforcement Learning.

TAL Traitement Automatique des Langues, en anglais Natural Language Processing (NLP).

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Systèmes de dialogue apprenant tout au long de leur vie : de l'élaboration à l'évaluation

Mots clés : systèmes de dialogue orientés tâche, lifelong learning, compréhension de la langue

Résumé : Les systèmes de dialogue orientés tâche, plus communément appelés chatbots, ont pour but de réaliser des tâches et de fournir des informations à la demande d'un utilisateur dans le cadre d'une conversation et d'un domaine précis (e.g. réservation d'un billet de train). Ces systèmes ont été largement adoptés par de nombreuses entreprises. Cependant, ils souffrent en pratique de certaines limitations : (1) ils sont dépendants des données d'entraînement nécessaires afin d'obtenir un système performant, (2) ils manquent de flexibilité et sont peu performants dès que le cas de figure rencontré en pratique s'éloigne des données vues pendant le développement, et (3) il est difficile de les adapter au cours du temps aux nouveaux éléments qui apparaissent étant donné l'évolution inévitable du monde et des exigences des concepteurs et des utilisateurs. Ainsi, nous appliquons le Lifelong Learning (LL) aux systèmes de dialogue orientés tâche. Nous définissons le LL comme la capacité d'un système à être appliqué à et à apprendre plusieurs tâches au cours du temps, en production, en autonomie, en continu et de manière interactive. Trois étapes doivent alors être réalisées en autonomie par le système : (1) Détecter la présence d'un nouvel élément, (2) extraire et identifier le nouvel élément et (3) adapter les composants du système associés à cet élément. Dans le cadre de cette thèse et étant donné la complexité du sujet, nous nous concentrons sur trois sous-problèmes liés aux systèmes de dialogue apprenant tout au long de leur vie.

Dans un 1er temps, nous proposons une 1ère méthodologie pour l'évaluation continue et au cours du temps de l'apprentissage sur le terrain des systèmes de dialogue. Ce type d'apprentissage est proche du LL mais met de côté l'aspect multi-tâches. Nous décrivons aussi un système de dialogue orienté tâche capable d'améliorer sur le terrain sa détection des slots via l'annotation autonome de données collectées au cours de ses interactions. Nous évaluons ce système à travers deux méthodes d'adaptation grâce à notre méthodologie et montrons l'intérêt d'une évaluation continue et au cours du temps.

Dans un 2nd temps, nous nous concentrons sur l'étude novatrice du transfert inter-langue dans le cadre de l'apprentissage continu d'une séquence de langues. En effet, le transfert et l'apprentissage continu sont deux aspects importants du LL. Nous réalisons cette étude sur la tâche de détection des slots à l'aide de BERT multilingue. Nous observons des capacités de transfert en avant substantielles malgré la présence d'oubli et présentons les capacités d'un modèle entraîné de manière continue.

Dans un 3ème temps, nous nous intéressons à l'étude du transfert inter-domaine dans le cadre de l'apprentissage zero-shot. Nous réalisons cette étude sur la tâche de suivi de l'état du dialogue, qui nécessite de considérer l'ensemble du dialogue et plus seulement le tour courant. Nous étudions d'abord les capacités de généralisation et de transfert d'un modèle existant sur de nouvelles valeurs de slots. Ensuite, nous proposons des variantes du modèle et une méthode capable d'améliorer les performances zero-shot du modèle sur des nouveaux types de slots appartenant à un nouveau domaine.

Title: Lifelong Learning Dialogue Systems: from Conception to Evaluation.....

Keywords: task oriented dialogue systems, lifelong learning, natural language understanding

Abstract: Task-oriented dialogue systems, more commonly known as chatbots, are intended to perform tasks and provide the information required by a user in a conversation in a specific domain (e.g., train booking). These systems have been widely adopted by many companies. However, they suffer in practice from some limitations: (1) they are dependent on the training data needed to obtain a performing system, (2) they lack flexibility and perform poorly as soon as the case encountered in practice moves away from the data seen during development, and (3) it is difficult to adapt them over time to new elements that appear given the inevitable evolution of the world, of the requirements of the designers and users. Thus we apply Lifelong Learning (LL) to task-oriented dialogue systems. We define LL as the ability of a system to be applied to and learn multiple tasks over time, in production, autonomously, continuously, and interactively. Three steps must be performed in autonomy by the system: (1) Detect the presence of a new element, (2) extract and identify the new element, and (3) adapt the system components associated with this element. As part of this thesis and given the complexity of LL, we focus our work on three subproblems associated with LL dialogue systems.

As a first step, we propose a first methodology for the continuous and time-dependent evaluation of on-the-job learning dialogue systems. This type of learning is close to LL but puts aside the multi-task aspect. We also describe a task-oriented dialogue system capable of improving its slot detection on-the-job via the autonomous annotation of data collected during its interactions. We evaluate this system through two adaptation methods using our methodology and show interest in a continuous evaluation over time.

As a second step, we focus on the innovative study of interlingual transfer when applying continual learning to a language sequence. Indeed, transfer and continual learning are two main aspects of LL. We perform this study on the slot-filling task using multilingual BERT. We observe substantial forward transfer capabilities despite the presence of forgetting and demonstrate the capabilities of a model trained in a continual manner.

As a third step, we study inter-domain transfer in the context of zero-shot learning. We carry out this study on a task that requires considering the whole dialogue and not only the current turn, which corresponds to the dialogue state tracking task. We first study the generalization and transfer capabilities of an existing model on new slot values. Then, we propose some model variants and a method able to improve the zero-shot performance of the model on new types of slots belonging to a new domain.