Vers la mitigation des biais en traitement neuronal des langues

Guillaume Le Berre

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Vers la mitigation des biais en traitement neuronal des langues

THÈSE

présentée et soutenue publiquement le 2 juin 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Guillaume Le Berre

Composition du jury

Président :	Armelle Brun
Rapporteurs :	Pascale Sébillot Benoit Favre
Examinateur :	Bang Liu
Directeurs de recherche :	Christophe Cerisara Philippe Langlais

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

Il est de notoriété que les modèles d'apprentissage profond sont sensibles aux biais qui peuvent être présents dans les données utilisées pour l'apprentissage. Ces biais qui peuvent être définis comme de l'information inutile ou préjudiciable pour la tâche considérée, peuvent être de différentes natures : on peut par exemple trouver des biais dans les styles d'écriture utilisés, mais aussi des biais bien plus problématiques portant sur le sexe ou l'origine ethnique des individus. Ces biais peuvent provenir de différentes sources, comme des annotateurs ayant créé les bases de données, ou bien du processus d'annotation lui-même. Ma thèse a pour sujet l'étude de ces biais et, en particulier, s'organise autour de la mitigation des effets des biais sur l'apprentissage des modèles de Traitement Automatique des Langues (TAL). J'ai notamment beaucoup travaillé avec les modèles pré-entraînés comme BERT, RoBERTa ou UnifiedQA qui sont devenus incontournables ces dernières années dans tous les domaines du TAL et qui, malgré leur large pré-entraînement, sont très sensibles à ces problèmes de biais.

Ma thèse s'organise en trois volets, chacun présentant une façon différente de gérer les biais présents dans les données. Le premier volet présente une méthode permettant d'utiliser les biais présents dans une base de données de résumé automatique afin d'augmenter la variabilité et la contrôlabilité des résumés générés. Puis, dans le deuxième volet, je m'intéresse à la génération automatique d'une base de données d'entraînement pour la tâche de question-réponse à choix multiples. L'intérêt d'une telle méthode de génération est qu'elle permet de ne pas faire appel à des annotateurs et donc d'éliminer les biais venant de ceux-ci dans les données. Finalement, je m'intéresse à l'entraînement d'un modèle multitâche pour la reconnaissance optique de texte. Je montre dans ce dernier volet qu'il est possible d'augmenter les performances de nos modèles en utilisant différents types de données (manuscrites et tapuscrites) lors de leur entraînement.

Abstract

It is well known that deep learning models are sensitive to biases that may be present in the data used for training. These biases, which can be defined as useless or detrimental information for the task in question, can be of different kinds: one can, for example, find biases in the writing styles used, but also much more problematic biases relating to the sex or ethnic origin of individuals. These biases can come from different sources, such as annotators who created the databases, or from the annotation process itself. My thesis deals with the study of these biases and, in particular, is organized around the mitigation of the effects of biases on the training of Natural Language Processing (NLP) models. In particular, I have worked a lot with pre-trained models such as BERT, RoBERTa or UnifiedQA which have become essential in recent years in all areas of NLP and which, despite their extensive pre-training, are very sensitive to these bias problems.

My thesis is organized in three parts, each presenting a different way of managing the biases present in the data. The first part presents a method allowing to use the biases present in an automatic summary database in order to increase the variability and the controllability of the generated summaries. Then, in the second part, I am interested in the automatic generation of a training dataset for the multiple-choice question-answering task. The advantage of such a generation method is that it makes it possible not to call on annotators and therefore to eliminate the biases coming from them in the data. Finally, I am interested in training a multitasking model for optical text recognition. I show in this last part that it is possible to increase the performance of our models by using different types of data (handwritten and typed) during their training.

Table des matières

Liste des tableaux

Table des figures

Chapitre 4. Génération automatique de bases de données débiaisées

Chapitre 5. Modèle multi-tâche pour mitiger les effets des biais

Remerciements

Je souhaite bien évidemment remercier mes deux co-superviseurs, Christophe Cerisara et Philippe Langlais, sans qui cette thèse n'aurait jamais été possible. Ce sont leur soutien, leurs conseils et les nombreuses heures passées à corriger mon anglais douteux qui m'ont permis de complèter cette thèse.

J'aimerais aussi remercier tous les membres du RALI et de SYNALP que j'ai rencontrés pendant ma thèse pour les discussions productives ainsi que pour les activités qui l'étaient moins et pour tous les bons moments passés ensemble. Je remercie tout particulièrement Guy Lapalme pour son aide précieuse sur le volet question-réponse de cette thèse.

Je remercie également l'entreprise LexRock AI d'avoir financé une partie importante de ma thèse, de m'avoir plongé dans le domaine de l'OCR et pour les interactions toujours constructives que nous avons pu avoir lors de nos réunions.

Chapitre 1

Introduction

Cette thèse, réalisée en cotutelle entre l'Université de Lorraine (Nancy, France) et l'Université de Montréal (Montréal, Canada) et sous la supervision de Christophe Cerisara (UL) et Philippe Langlais (UdeM), a pour sujet l'étude et la mitigation des biais en apprentissage machine. Je m'intéresse pour cela à divers domaines du Traitement Automatique des Langues (TAL) : le question-réponse, le résumé automatique, et la reconnaissance optique de caractères. Chacune de ces tâches, me permet de mettre en lumière différents aspects des biais et diverses façons de s'en défaire.

Dans cette thèse, je m'intéresse tout particulièrement à une sous-catégorie de l'apprentissage automatique : les réseaux de neurones et l'apprentissage profond. Cette discipline consiste à l'entraînement d'un modèle constitué d'un ensemble d'opérateurs non-linéaires simples appelés neurones (souvent constitués d'un opérateur linaire suivi d'une fonction d'activation non-linaire). Ces neurones qui par eux-mêmes ne peuvent pas représenter d'information complexe dû à leur simplicité, sont organisés dans des modèles d'architecture élaborée qui eux, sont capables de traiter des informations structurées. Ces modèles sont entraînés, souvent sur une tâche spécifique, avec différentes méthodes de descente de gradient et peuvent à terme être utilisés pour résoudre la tâche sur laquelle ils ont été entraînés avec différents degrés d'efficacité. Ce champ de recherche est complexe et je ne peux pas ici en faire une présentation exhaustive mais de nombreux ouvrages en présentent déjà les subtilités (Goodfellow, Bengio et Courville 2016).

Ces méthodes d'apprentissage profond qui commencent surtout à devenir populaires au début des années 2000, ont montré dans de nombreux domaines des résultats bien supérieurs

à ce qui était utilisé jusqu'alors. Les premières applications à succès, ce sont concentrées sur le domaine du traitement d'image (LeCun et al. 1998 ; Lu et Weng 2007) mais la discipline s'est rapidement étendue à d'autres domaines tels que le TAL ou le traitement de la parole. Malgré leur succès et leur adoption dans de nombreux domaines industriels, ces méthodes souffrent de plusieurs problèmes limitant leur utilisation dans des domaines plus critiques ou à plus fort impact. Parmi ces problèmes, on peut mentionner la nécessité d'avoir accès à une grande quantité de données annotées, ce qui rend l'entraînement de tels modèles difficile dans certains domaines où l'annotation est plus coûteuse en temps et en main d'œuvre. C'est cette première limitation qui a inspiré ces dernières années de nombreux articles scientifiques sur le sujet de l'apprentissage faiblement supervisé ou non supervisé. C'est une autre de ces limitations qui va nous intéresser dans la suite de cet ouvrage : le problème des biais. Ces biais qui, à terme, se retrouvent dans les modèles vont alors entraîner des prédictions erronées qui auront probablement peu de répercussion dans le cas par exemple d'un modèle de traduction automatique mais sont beaucoup plus problématiques lorsqu'on prévoit une utilisation dans des applications reliées à la justice ou la conduite automatique.

1.1. Les biais en apprentissage machine

Dans cette section, je commence par introduire ce qu'est un biais et quels sont les principaux types de biais que l'on peut retrouver dans des applications d'apprentissage profond. Je discute ensuite de certaines problèmatiques liées aux biais en apprentissage machine afin de donner au lecteur une idée plus précise de l'écosystème général dans lequel mes travaux s'inscrivent. Cette étude n'est bien évidemment pas exhaustive tant le sujet des biais en apprentissage machine est vaste.

1.1.1. Définition et classification des biais

Tout d'abord, il est nécessaire de définir ce qu'est un biais. Un biais peut être défini comme une information contenue dans une base de données ou un modèle qui est inutile ou pénalisante pour une tâche donnée. Le biais est donc une notion qui est relative à une tâche ou une utilisation donnée et ce qui est un biais pour une tâche peut être une information utile pour une autre. Cette observation explique d'ailleurs en partie pourquoi le biais

est si problématique en apprentissage automatique puisque l'identification même d'un biais nécessite de prendre en compte la tâche et le contexte.

En apprentissage profond, les biais peuvent venir de différentes sources et de nombreux articles scientifiques et revues de littérature proposent diverses classifications des différents types de biais (Mehrabi et al. 2021 ; Suresh et Guttag 2019 ; Olteanu et al. 2019). Mehrabi et al. (2021) définit 3 catégories de biais : les biais imputables aux personnes, ceux imputables au processus d'annotation et ceux imputables aux modèles et aux algorithmes.

Premièrement, une partie des biais est imputable aux personnes elles-mêmes. Ces biais sont introduits par les personnes ayant annoté les bases de données, les personnes responsables de la création du protocole de récolte des données ou bien la population sur laquelle les données portent. Ce type de biais inclut entre autres :

— Des biais historiques qui sont des biais déjà présents dans la population. Mehrabi et al. (2021) donnent l'exemple de la proportion débalancée de femmes à la tête de grandes entreprises. Dans le même thème, Stanovsky, Smith et Zettlemoyer (2019) montrent que certains modèles de traduction automatique ont tendance à faussement attribuer certains genres (masculin ou féminin) en fonction de certaines professions lorsque le modèle traduit d'un langage faiblement genré (ex. l'anglais) vers un langage plus fortement genré (ex. l'espagnol ou le français).

— Des biais de population qui apparaissent lorsque que la population sur laquelle est créée la base de données est différente de la population cible que l'on voudrait représenter. On peut prendre l'exemple d'une base de données de texte construite uniquement à partir d'un ensemble d'articles scientifiques. Une telle base de données va alors contenir un vocabulaire très spécifique qui n'est pas forcément représentatif du vocabulaire commun.

— Des biais temporels qui résultent de la création d'une base de données avec des données d'une autre époque. Ces biais sont similaires aux biais de population et vont résulter en une base de données qui ne sera pas forcément représentative de la situation actuelle.

Ensuite, certains biais peuvent apparaître au moment de la création des bases de données. Ces biais résultent de la manière avec laquelle les données sont collectées. Cette catégorie de biais inclut :

— Des biais de mesure. Ces biais dépendent de la manière avec laquelle la mesure d'une donnée est effectuée, comme par exemple lorsque l'on effectue une mesure sur un proxi en lieu et place de la donnée réelle. Ce proxi peut alors être biaisé et ne pas être complètement corrélé à la donnée réelle que l'on cherche à mesurer.

— Des biais de représentation, qui résultent d'une collecte des données sur une partie d'une population qui peut ne pas être représentative de la population globale. Ce biais est assez fréquent dans les bases de données d'apprentissage profond puisque la majeure partie des bases de données sont collectées dans un petit ensemble de pays (Amérique du nord et Europe principalement) et ont donc tendance à sous-représenter certains langages ou origines ethniques.

— Des biais d'agrégation. Ces biais résultent de l'amalgame d'une population dans une base de données servant ensuite à prendre des décisions qui concernent des individus ou des sous-groupes d'individus. Or, une information qui peut être vraie en moyenne sur une population peut ne pas s'appliquer à un individu particulier. Ce genre de biais pose des problèmes éthiques importants dans certains cas, comme l'utilisation de modèles d'apprentissage profond dans le domaine de la justice. Par exemple, une probabilité de récidive calculée sur une population ne justifie en rien que cette probabilité s'applique à un individu donné.

Enfin, certains biais peuvent apparaître au moment de l'apprentissage des modèles ou la diffusion de ceux-ci aux utilisateurs. Ce type de biais inclut :

— Des biais algorithmiques. Ces biais sont amenés par la manière dont les modèles sont entraînés alors même que les données utilisées pour l'apprentissage ne sont pas forcément biaisées. Ce genre de biais est particulièrement problématique pour les modèles d'apprentissage profond puisque ceux-ci sont des boites noires et qu'il est difficile de savoir exactement ce qu'ils ont appris.

— Des biais de popularité qui résultent du fait que les modèles et algorithmes les plus populaires ne sont pas toujours les meilleurs.

— Des biais d'évaluation qui résultent de l'évaluation des modèles et donc de la manière avec laquelle on sélectionne quel modèle est le meilleur. Ces biais peuvent être dus par exemple à la métrique utilisée pour l'évaluation comme dans le cas du résumé automatique où l'une des mesures les plus utilisées – ROUGE (C.-Y. Lin 2004) – ne

mesure pas vraiment la qualité d'un résumé mais la correspondance mot à mot avec un résumé cible arbitraire.

1.1.2. Rapide tour d'horizon du biais en apprentissage machine

Les biais sont un problème très présent dans le domaine de l'apprentissage profond et touchent toutes ses sous-disciplines. On peut mentionner par exemple la classification d'images où il a été depuis longtemps démontré que certains modèles apprennent à reconnaitre des objets en fonction du contexte dans lequel ils apparaissent (K. K. Singh et al. 2020 ; Z. Zhu, Xie et Yuille 2017). Un avion ou un oiseau a par exemple beaucoup plus de chance de se trouver sur un fond bleu. Cette corrélation est intéressante dans certains cas puisqu'elle peut permettre aux modèles de conforter une décision lorsque l'objet étudié est ambigu. En revanche si un poids trop important est mis sur le contexte, cela peut gêner la détection et amener à une mauvaise classification d'un oiseau posé au sol. Puisque les modèles d'apprentissage profond sont encore principalement des boites noires et qu'il est très difficile de comprendre ce qu'ils considèrent pour prendre telle ou telle décision, ces problèmes associés aux biais réduisent la confiance que l'on peut leur accorder et bloquent encore aujourd'hui leur application à des domaines sensibles. Cette problématique pousse de plus en plus de chercheurs à tenter d'expliquer le comportement des modèles de réseau de neurones (Ribeiro, S. Singh et Guestrin 2016 ; Van Aken et al. 2019).

Le TAL ne fait pas exception. De nombreux travaux portent par exemple sur certains biais historiques comme des biais racistes ou sexistes. On peut reparler ici par exemple du cas de la traduction automatique impactée par la présence de biais sur le genre (Stanovsky, Smith et Zettlemoyer 2019 ; Prates, Avelar et Lamb 2020). On retrouve aussi en TAL de nombreuses bases de données qui contiennent des biais d'annotation. C'est le cas par exemple pour le NLI (*Natural Language Inference*) (Levy et al. 2015 ; Tsuchiya 2018 ; Gururangan et al. 2018) ou, comme présenté dans le chapitre 2, pour le question-réponse. Ces biais d'annotation sont parfois entremêlés à des biais imputables aux différents styles d'écriture de chaque annotateurs qui permettent dans certains cas à des modèles de retrouver et d'utiliser l'identité des annotateurs (Geva, Goldberg et Berant 2019).

1.2. Les modèles pré-entraînés

Récemment, un grand nombre de modèles pré-entraînés ont vu le jour. Ces modèles sont rapidement devenus incontournables dans un grand nombre de tâches de TAL et notamment pour la tâche de question-réponse où ils occupent l'intégralité de l'état de l'art actuel. Comme la réalisation de cette thèse s'est déroulée simultanément avec l'apparition et la montée en puissance de ces modèles pré-entraînés, j'ai eu l'occasion de beaucoup travailler avec eux. Comme j'utilise plusieurs d'entre eux dans les chapitres qui vont suivre, il me semble important de détailler ici les modèles pré-entraînés les plus importants.

Cette section présente donc les principaux modèles pré-entraînés utilisés pour le question-réponse en commençant par le plus ancien d'entre eux : BERT (Devlin et al. 2019). Cette liste n'est pas exhaustive et ne présente pas les dizaines de modèles existants. On peut citer par exemple Reformer (Kitaev, Kaiser et Levskaya 2020) ou XLNet (Yang, Dai et al. 2019) qui proposent des améliorations structurelles à BERT. Ces modèles pré-entraînés sont d'ailleurs désormais disponibles dans différentes langues comme le français avec FlauBERT (Le et al. 2020), CamemBERT (Martin et al. 2020), BARThez (Kamal Eddine, Tixier et Vazirgiannis 2021) ou encore GPT_{fr} (Simoulin et Crabbé 2021).

1.2.1. BERT

BERT (Devlin et al. 2019) est le tout premier modèle pré-entraîné de son type. Ce modèle permet d'obtenir des vecteurs de représentation des mots dans une phrase. Contrairement à beaucoup de modèles ou méthodes précédemment utilisés, les vecteurs de représentation produits par BERT sont dépendants du reste des mots (à la fois les mots précédents et suivants) dans la phrase et incorporent donc une information sur le contexte. BERT utilise une architecture de type *transformer* (Vaswani et al. 2017). Cette architecture comprend entre autres plusieurs couches d'attention permettant de compresser l'information du reste de la phrase dans la représentation d'un mot. Les phrases sont complétées par une série de *tokens* spéciaux comme le token " [CLS] " pour marquer le début d'une phrase. BERT modélise les mots dans une phrase à l'aide de sous-mots, selon la méthode *WordPiece* décrite dans (Y. Wu et al. 2016). De plus, BERT peut prendre en entrée un couple de phrases (utilisé pour le couple question/réponse par exemple et utile pour la deuxième tâche de pré-entraînement présentée ci-après) et une représentation supplémentaire est ajoutée pour encoder de quelle

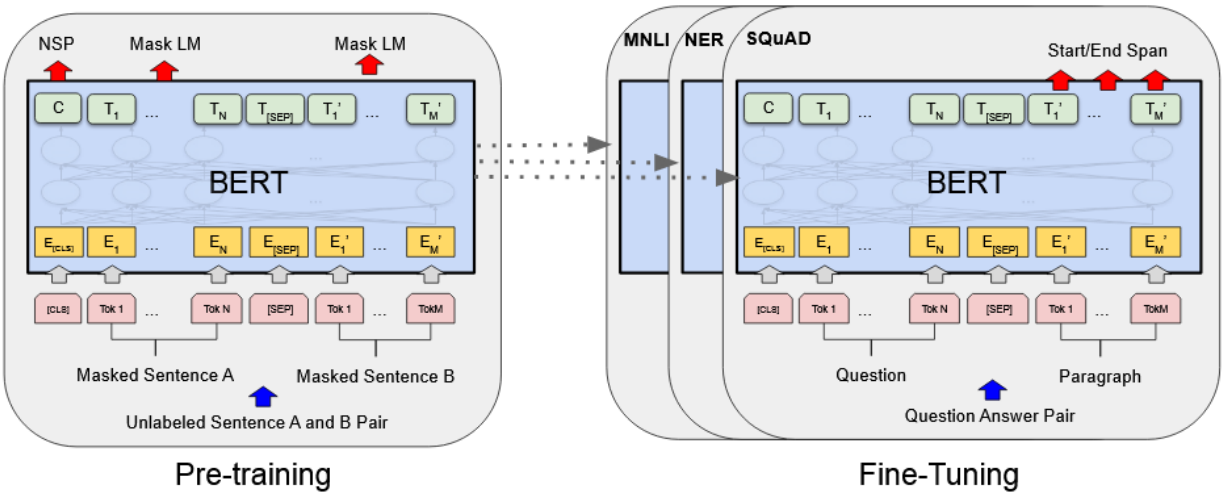


Figure 1 – Schema du pré-entraînement et de l'architecture de BERT. Image provenant de Devlin et al. (2019).

phrase A ou B un mot est issu. BERT est pré-entraîné de manière non-supervisée sur 2 tâches :

— Une tâche où le modèle est entraîné à retrouver des mots manquants (remplacé par un token spécial " [MASK] "). Le modèle doit utiliser les mots qui entourent ce token masqué et donc doit encoder une partie du contexte (le plus pertinent).

— Pour la deuxième tâche, on sélectionne 2 phrases dans une base de données non-annotée. L'objectif du modèle dans cette configuration est de déterminer si les deux phrases qu'on lui donne se suivent ou non dans le texte initial. L'idée est de forcer le modèle à encoder de la sémantique dans ces représentations puisque celui-ci doit comprendre le contenu des phrases (ou au moins reconnaitre le champ lexical) pour pouvoir répondre correctement. Pour cette tâche, c'est le vecteur de représentation du premier token (" [CLS] ") qui est utilisé et une couche linéaire est ajoutée pour faire la prédiction.

Le pré-entraînement de BERT est effectué en utilisant deux gros corpus non-annotés : BooksCorpus (800 millions de mots) (Y. Zhu et al. 2015) et Wikipédia en anglais (2 500 millions de mots). BERT est disponible en 2 versions variant sur leur taille et donc sur leur capacité :

— Une version "Base" de 110 millions de paramètres avec 12 niveaux d'attention avec chacun 12 têtes d'attention et donnant des représentations de dimension 768.

— Une version “Large” de 336 millions de paramètres, 24 niveaux, 16 têtes et des représentations de dimension 1024.

BERT est initialement pré-entraîné pour l'anglais mais une version multilingue à aussi été publiée. De plus, depuis l'introduction de BERT, de nombreuses autres versions pour d'autres langues ont été proposées comme par exemple pour le français (Martin et al. 2020) ou l'arabe (Ghaddar et al. 2022).

1.2.2. RoBERTa

RoBERTa (Robustly Optimized BERT Approach) (Y. Liu et al. 2019) est une amélioration de BERT où la différence principale réside dans l'utilisation de la première tâche de pré-entraînement uniquement (tâche de récupération des mots masqués) et dans le recours à l'encodage BPE (Byte-Pair Encoding) (Sennrich, Haddow et Birch 2016) plutôt que WordPiece. De plus, le masquage des mots utilisé par BERT est statique, c'est à dire que les mots sont masqués une fois au moment de préparer les données et tous les passages sur les données lors de l'apprentissage se font avec le même masquage. RoBERTa, à l'inverse, est entraîné en refaisant le masquage à chaque nouveau passage sur les données, ce qui permet d'augmenter de manière artificielle la taille du corpus d'entraînement. Finalement, RoBERTa est entraîné avec plus de données pendant plus longtemps et en utilisant des *mini-batchs* plus grandes.

1.2.3. ALBERT

ALBERT (Lan et al. 2020) est un autre modèle inspiré de BERT utilisant moins de ressources. Une des premières observations faites dans (Lan et al. 2020) est que dans les modèles précédents, les vecteurs *one-hots* encodant les tokens étaient directement projetés sur un vecteur de dimension correspondant à la dimension des vecteurs de représentation souhaités en sortie. Cette approche est la plus simple à mettre en place mais n'est pas vraiment optimale puisque qu'a priori l'information contenue dans un token devrait être de moindre dimension par rapport aux vecteurs de représentation en sortie dans la mesure où ces derniers encodent en plus l'information du contexte. De plus, cette façon de faire est coûteuse surtout si le vocabulaire utilisé est grand et engendre un coût de l'ordre de $V \times H$ paramètres (avec V la taille du vocabulaire et H la dimension des vecteurs de représentations). Les créateurs

d'ALBERT proposent donc d'abord de projeter les mots sur des vecteurs de taille $E < H$ puis d'étendre la taille de ce vecteur. Ainsi le nombre de paramètres diminue pour atteindre l'ordre de grandeur de $V \times E + E \times H$. De plus, ALBERT optimise le nombre de paramètres en partageant tous les paramètres entre les différents niveaux du réseau "transformer". Ces deux optimisations permettent d'obtenir un réseau beaucoup plus petit en termes de nombre de paramètres par rapport aux alternatives. Du côté de la méthode d'entraînement, la différence majeure avec BERT est que pour ALBERT la tâche de prédiction de la phrase suivante est modifiée. Dans BERT, les exemples positifs sont sélectionnés en prenant des phrases qui se suivent dans le texte et les exemples négatifs sont créés en prenant deux phrases dans deux documents différents. Cette approche a tendance à pousser le modèle à apprendre à reconnaitre des similarités de champ lexical entre les phrases et pas vraiment à comprendre le sens des phrases en question puisque les exemples négatifs ont peu de chance d'avoir ne serait-ce que le même thème. C'est pour cela que lors du préapprentissage d'ALBERT, les exemples négatifs sont créés en prenant deux phrases se suivant dans le texte et en les inversant. Ainsi, les thèmes abordés dans les deux phrases sont toujours les mêmes et il devient plus important de réellement comprendre ce qui est dit.

1.2.4. T5

T5 (Raffel et al. 2020) est un modèle encodeur-décodeur basé sur une architecture de type "transformer". L'idée avec T5 est de proposer un seul modèle pré-entraîné pouvant servir de base à toutes les tâches de texte à texte comme la traduction, le résumé de texte, le question-réponse ou encore l'analyse de sentiment. De plus, certaines tâches qui ne sont pas au premier abord des tâches texte à texte comme par exemple SNLI (Bowman et al. 2015) peuvent être modifiées pour le devenir. SNLI rassemble 570 000 paires de phrases avec des annotations indiquant si les phrases en question se complémentent (*entailment*), se contredisent (*contradiction*) ou n'ont aucune relation l'une avec l'autre (*neutral*). Il est possible de transformer SNLI en une tâche texte à texte en ayant d'un côté les deux phrases concaténées et de l'autre une séquence constituée d'un seul mot ("*entailment*", "*contradiction*" ou "*neutral*"). L'idée avec T5 est donc de rassembler toutes les tâches de TAL en une seule et même tâche texte à texte.

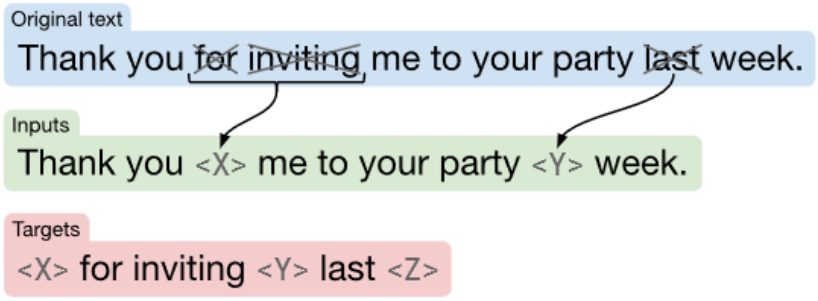


Figure 2 – Exemple de génération non-supervisée des entrées et sorties utilisées pour le pré-entraînement du modèle T5. Des morceaux de textes sont retirés aléatoirement dans une phrase et le modèle est entraîné à retrouver et générer les bouts manquants. Image provenant de Raffel et al. (2020).

Le pré-entraînement de T5 est fait sur un grand nombre de textes automatiquement extraits d'internet et suivant une méthode qui se rapproche du masquage de mots utilisée par BERT. 15% des mots sont supprimés de manière aléatoire et chaque séquence de mots consécutifs supprimés est remplacée par un token parmi une sélection de tokens spéciaux avec un identifiant unique. La phrase objectif est ensuite créée grâce aux mots masqués dans la phrase en entrée séparés par les mêmes tokens spéciaux. La figure 2 présente plus clairement le type de séquences utilisées pour le pré-entraînement de T5. Ce modèle est disponible en 5 versions : "Small" (∼60 millions de paramètres), "Base" (∼220 millions), "Large" (∼770 millions), "3B" (∼2.8 milliards) et "11B" (∼11 milliards).

1.2.5. GPT

GPT (Radford et Narasimhan 2018) est un modèle de langue développé par une équipe de recherche d'OpenAI [1]. GPT est basé sur un décodeur *transformer*. Il est donc composé en plus d'une couche d'*embedding* suivi de 12 couches transformer constituées chacune d'une couche de *self-attention* (avec 12 têtes d'attention) et d'une couche linéaire (*position wise*). GPT utilise des vecteurs de représentation de dimension 768.

Le modèle est d'abord entraîné de manière non-supervisée en utilisant la base de données BookCorpus (Y. Zhu et al. 2015) (utilisée également pour l'entraînement de BERT). Cet entraînement est réalisé de la manière classique pour les modèles de langue, c'est à dire que

1. `https://openai.com/`

le modèle est entraîné à prédire le token suivant dans une phrase en fonction des tokens précédents.

Bien que GPT soit entraîné comme un modèle de langue, son utilité s'étend bien au délà de cette simple utilisation. En effet, la contribution principale apportée par (Radford et Narasimhan 2018) est d'avoir montré que ce modèle pouvait être utilisé (avec un affinage adéquat) pour de nombreuses tâches allant de la classification au question-réponse avec dans chacune de ces tâches des résultats supérieurs aux autres modèles disponibles au moment. Pour la tâche de classification par exemple, il convient de prendre le dernier vecteur d'état (celui correspondant au dernier token de la phrase) et de lui appliquer une ou plusieurs couches linéaires permettant de faire la prédiction des classes. On peut ensuite affiner le modèle avec cette architecture sur une base de données de classification.

GPT a eu pour l'heure deux successeurs. GPT-2 (Radford, J. Wu et al. 2019) est une version plus large ($\sim$1.5 milliards de paramêtres) entraîné sur un nouveau corpus : WebText. WebText est un corpus constitué de 45 millions de liens vers des pages webs sélectionnées à partir de Reddit. Allant encore plus loin que son prédécesseur GPT, GPT-2 est capable, une fois entraîné de manière non-supervisée sur WebText, d'obtenir de bons résultats sur d'autre tâches (par exemple du résumé automatique ou de la traduction) sans avoir besoin d'un affinage spécifique sur cette tâche. Pour générer une traduction par exemple, on fournit au modèle une seule longue séquence de mots contenant d'abord quelques exemples de traductions sous la forme `phrase en anglais = phrase en français`, puis un dernier exemple incomplet `phrase en anglais =`. On laisse ensuite le modèle compléter lui-même l'exemple. Radford, J. Wu et al. (2019) montrent donc qu'avec un pré-entraînement sur une base de données suffisamment diverse, il est possible pour un modèle d'aprendre à reproduire un comportement (comme traduire une phrase) à partir d'exemples (mais sans entraînement additionnel). Finalement, GPT-3 (Brown et al. 2020) est une version encore plus large de GPT-2 ($\sim$ 175 milliards de paramètres) mais similaire à ce dernier dans sa structure et son entraînement.

1.3. Plan de la thèse

Dans cette thèse que je présente dans les quatre chapitres qui suivent, je m'intéresse à différentes manières de gérer les biais que l'on peut trouver en Traitement Automatique

des Langues lorsqu'on utilise des modèles neuronaux. Je m'intéresse principalement aux biais qui apparaissent lors de la création, et donc de l'annotation, des bases de données en question. Comme ma thèse a été financée à partir de différentes sources, chacune ayant des conditions spécifiques, j'ai eu l'occasion de travailler sur plusieurs tâches du Traitement Automatique des Langues. Ces tâches qui sont : le question-réponse, le résumé automatique et la reconnaissance optique de texte, serviront donc d'illustrations à mes travaux. Comme j'ai travaillé sur des domaines variés et afin d'épargner au lecteur une introduction trop longue, j'introduirai les concepts spécifiques à chaque tâche au début des chapitres concernés.

Dans le chapitre 2, je commencerai par illustrer les problèmes causés par les biais d'annotation sur une tâche de question-réponse à choix multiples. Ces premières expériences, réalisées au début de mon doctorat, m'ont permis de me rendre compte de l'impact des biais sur l'apprentissage et les résultats des modèles d'apprentissage profond. Dans cette première partie, je montre donc qu'il existe des biais importants dans certaines des bases de données les plus utilisées en question-réponse. De plus, je montre également que, contrairement à ce qu'on pourrait penser, les modèles pré-entraînés de type BERT ou RoBERTa qui venaient alors tout juste d'émerger ne sont pas insensibles à ces biais. En effet, malgré leur large pré-entraînement qui supposément pourrait permettre de balancer les éventuels biais présents dans la base de données d'affinage, ces modèles sont prompts à intégrer ces biais. Ce sont ces premières expériences qui m'ont mené à travailler sur des méthodes d'atténuation des biais. Ma thèse s'organise en trois volets explorant respectivement des méthodes permettant d'utiliser, d'éliminer et de mitiger les biais présents dans les bases de données.

Dans le chapitre 3, je m'intéresse à la tâche de résumé automatique. Un problème majeur pour cette tâche (et qui se retrouve d'ailleurs dans d'autres tâches du TAL) est la présence dans les bases de données de différentes façons d'écrire. En effet, il n'existe pas une seule manière de résumer un texte et chaque annotateur a son propre style d'écriture. Par exemple, certains annotateurs vont faire des résumés plus longs en incluant plus d'information alors que d'autres iront droit au but et produiront donc des résumés plus courts. Ce phénomène est problématique pour l'apprentissage avec des méthodes de descente de gradient puisque les modèles ne peuvent pas converger vers une représentation qui colle à l'ensemble des exemples d'entraînement. Cette variabilité peut de plus être considérée comme un biais propre à chaque annotateur et chaque exemple d'entraînement est ainsi biaisé par le style

d'écriture de l'annotateur qui l'a produit. Dans cette partie, je présente une architecture et une méthode d'apprentissage qui permet de capturer cette variabilité présente dans la base de données de manière non supervisée (sans donner explicitement au modèle d'information sur les annotateurs). Notre modèle est ensuite capable de produire, pour chaque phrase d'entrée, plusieurs résumés avec différents styles. Les biais deviennent alors une information utile dont on peut se servir plutôt qu'une gêne pour l'apprentissage des modèles.

Ensuite, dans le chapitre 4, je discute de l'élimination des biais directement à la source dans les bases de données via la création automatique non supervisée de données d'entraînement. Dans ce chapitre, je travaille de nouveau sur la tâche de question-réponse à choix multiples et je présente une méthode permettant de générer automatiquement des questions ainsi que les distracteurs (mauvaises réponses) associés à partir de phrases simples décrivant le domaine d'intérêt (par exemple issues de Wikipédia). L'idée derrière une telle méthode de génération est que l'on peut contrôler la manière avec laquelle on crée les questions de manière à éviter d'y introduire des biais. Ces données synthétiques peuvent alors permettre d'entraîner des modèles de manière complétement non supervisée et je montre dans cette partie que l'on peut déjà obtenir ainsi des résultats prometteurs. Elles peuvent aussi servir à affiner des modèles généraux déjà pré-entraînés, sur de nouvelles tâches ou domaines, à moindre coût d'annotation et en évitant d'apporter des biais dans l'apprentissage.

Enfin, dans le chapitre 5, je m'intéresse à l'entraînement d'un modèle multitâche pour la reconnaissance optique de texte. J'utilise donc un ensemble de données variées mélangeant des textes manuscrits et tapuscrits. Les intérêts de tels modèles sont multiples et des méthodes similaires ont montré des performances accrues sur d'autres tâches de TAL comme le question-réponse. D'une part, un modèle qui est entraîné sur plusieurs tâches semblables mais variées, va avoir une plus grande capacité de généralisation à de nouvelles tâches connexes. De plus, cela permet d'avoir un ensemble d'entraînement plus conséquent, ce qui est particulièrement intéressant en apprentissage profond sur des tâches où les données annotées sont difficiles à obtenir. Et finalement, cela permet de noyer les biais qui peuvent se trouver dans l'une ou l'autre des bases de données utilisées. En effet, il devient alors moins intéressant pour le modèle d'apprendre un biais se trouvant dans une seule ou un petit nombre des bases de données. Je montre dans ce chapitre qu'un modèle multitâche pour la reconnaissance de

texte obtient une augmentation significative de performance comparé à des modèles entraînés sur un seul type de données. Et cela se confirme même lorsqu'on évalue nos modèles sur des bases de données qui n'ont pas été vues lors de l'entraînement.

Chapitre 2

Mise en lumière de biais dans des bases de données de question-réponse à choix multiples

Dans ce chapitre, je mets en évidence des biais importants se trouvant dans certaines bases de données de question-réponse à choix multiples les plus utilisées. De plus, je montre que les modèles pré-entraînés et en particulier BERT ne sont pas immunisés à ces biais et que ceux-ci influencent largement l'affinage de ces modèles.

Dans la première partie de mon doctorat, j'ai travaillé sur des bases de données de questions-réponse et j'ai pu mettre en évidence plusieurs problèmes avec ces bases de données. Premièrement, la création de ce genre de base de données demande un travail d'annotation conséquent et les bases de données de question-réponse contiennent donc souvent peu de questions, surtout lorsqu'on cherche à utiliser ces questions pour entraîner des systèmes d'apprentissage profonds. De plus, j'ai pu mettre en évidence que certaines bases de données de questions à choix multiples comme OpenBookQA ou ARC contiennent des biais importants qui influencent l'apprentissage des modèles d'apprentissage profonds (y compris les modèles pré-entraînés). Ces biais qui sont au moins en partie liés à la manière d'annoter les données sont sans aucun doute également présents dans d'autres bases de données de question-réponse et rendent encore plus difficile la création de nouvelles bases. Le travail présenté dans ce chapitre a fait l'objet d'une publication à la conférence CanadianAI 2020 (Le Berre et Langlais 2020).

The English name “Normans” comes from the French words Normans/Normanz, plural of Normant, modern French normand, which is itself borrowed from Old Low Franconian Nortmann “Northman” or directly from Old Norse Norðmaðr, Latinized variously as Nortmannus, Normannus, or Nordmannus (recorded in Medieval Latin, **9th century**) to mean “**Norseman, Viking**”.
What is the original meaning of the word Norman ? - **Viking / Norseman, Viking**
When was the Latin version of the word Norman first recorded ? - **9th century**
When was the French version of the word Norman first recorded ? - <no answer>

Figure 3 – Un exemple de paragraphe et questions associées dans SQUAD2.0

2.1. Introduction

2.1.1. Introduction générale sur le question-réponse

Le question-réponse est depuis longtemps une tâche essentielle du Traitement Automatique des Langues (TAL). En raison de l'essor de l'apprentissage profond, on assiste depuis peu à une recrudescence des travaux sur le question-réponse, entraînant une multiplication des *benchmarks* (Bai et D. Z. Wang 2021). Malgré d'importants progrès réalisés par les méthodes d'apprentissage profonds, les modèles actuels ne parviennent pas encore à atteindre les performances humaines sur un grand nombre de tâches. Comme pour les autres tâches de TAL, le question-réponse comporte un large éventail de sous-tâches. Certains ensembles de données de question-réponse extractives fournissent une question ouverte sur un texte court et exigent que les modèles sélectionnent un segment du texte (correspondant souvent à une entité nommée) qui répond à la question. SQuAD1.1 (Rajpurkar, Zhang et al. 2016) (100 000 questions) et SQuAD2.0 (Rajpurkar, Jia et P. Liang 2018) (150 000 questions) sont deux exemples populaires de ce type de bases de données. La principale différence entre ces deux bases de données est que SQuAD2.0 contient, en plus des questions de SQuAD1.1, des questions pour lesquelles la réponse n'est pas présente dans le texte fourni. Les modèles doivent alors correctement estimer qu'il n'est pas possible de répondre à la question. La figure 3 présente quelques exemples de questions extraites de SQuAD2.0. D'autres bases de données de question-réponse extractives incluent NewsQA (Trischler et al. 2017) (100 000 questions), Quoref (Dasigi et al. 2019) (24 000 questions) et ROPES (K. Lin et al. 2019) (14 322 questions).

D'autre part, les bases de données abstractives comme CoQA (Reddy, Chen et Manning 2019) (∼127 000 questions), NaturalQuestions (Kwiatkowski et al. 2019) (∼300 000 questions), NarrativeQA (Kočiský et al. 2018) (46 765 questions), DROP (Dua et al. 2019) (96 000 questions) ou HotpotQA (Yang, Qi et al. 2018) (113 000 questions) demandent plutôt aux modèles de générer une réponse sans que celle-ci ne soit présente telle quelle dans un texte de référence. Ce type de questions demande alors au modèle plus de raisonnement pour analyser le ou les textes de référence si la base de données en contient (ex. HotPotQA), ou bien si la base de données ne contient pas de texte de référence (ex. NaturalQuestions), les modèles doivent baser leurs réponses sur une connaissance du monde apprise au préalable. L'évaluation des modèles sur ce type de questions est souvent plus difficile que dans le cas de bases de données extractives. En effet, les questions étant assez ouvertes, il est possible qu'un modèle génère une réponse sémantiquement correcte mais qui est très loin de la réponse attendue. Cette problématique est partagée avec d'autres tâches de TAL comme par exemple le résumé automatique où les métriques traditionnellement utilisées (ROUGE (C.-Y. Lin 2004) ou BLEU (Papineni et al. 2002)) ne parviennent pas à capturer la proximité sémantique entre un résumé généré et le résumé cible.

Dans d'autres bases de données de question-réponse à choix multiples, comme dans RACE (Lai et al. 2017) (97 687 questions), MCTest (Richardson, Burges et Renshaw 2013) (2 000 questions), CommonSenceQA (Talmor et al. 2019) (12 247 questions) ou QASC (Khot et al. 2020) (9 980 questions), plusieurs choix de réponses sont proposés aux modèles et ceux-ci doivent sélectionner la réponse correcte. La figure 4 montre des exemples de questions dans RACE. Un des avantages de ce type de question-réponse, est qu'il est facile d'évaluer objectivement les performances des modèles contrairement aux bases de données abstractives. De plus, tout comme ces dernières, les bases de données de questions à choix multiples demandent aux modèles d'avoir une bonne connaissance du monde ainsi qu'une bonne compréhension des textes de référence éventuels.

On peut aussi noter l'existence de certaines bases de données contenant des questions auxquelles il faut répondre par "Oui" ou par "Non". C'est le cas par exemple de BoolQ (C. Clark et al. 2019) (16 000 questions). Ce type de bases de données peut s'apparenter aux bases de données à choix multiples.

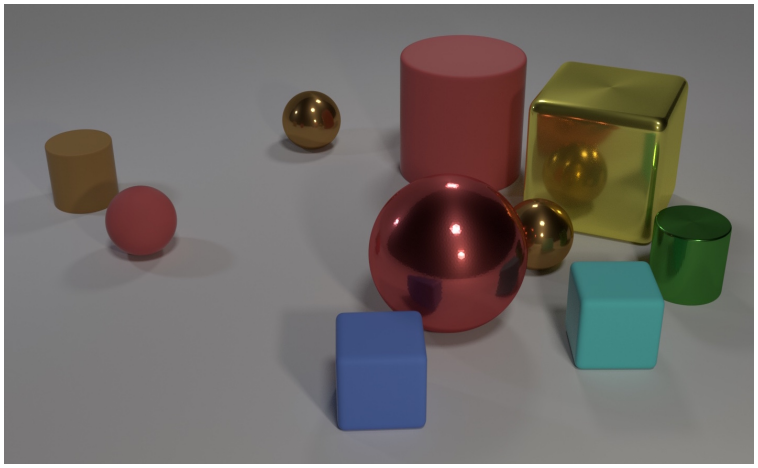
In a small village **in England** about 150 years ago, a mail coach was standing on the street. It didn't come to that village often.People had to pay a lot to get a letter. The person who sent the letter didn't have to pay the postage, while the receiver had to. "Here's a letter for Miss Alice Brown," said the mailman. "I'm Alice Brown," a girl of about 18 said in a low voice. Alice looked at the envelope for a minute, and then handed it back to the mailman. "I'm sorry I can't take it, **I don't have enough money to pay it**", she said. (...)
The first postage stamp was made _____. A) **in England** B) in America C) by Alice D) in 1910
The girl handed the letter back to the mailman because _____. A) she didn't know whose letter it was B) **she had no money for the postage** C) she received the letter but she didn't want to open it D) she had already known what was written in the letter

Figure 4 – Un exemple de paragraphe dans RACE avec 2 des questions associées

Finalement, certaines bases de données de question-réponse mélangent TAL et traitement de l'image en proposant des questions portant sur le contenu d'une image. C'est le cas par exemple pour la base de données CLEVR (Johnson et al. 2017) qui contient des questions portant sur des images synthétiques montrant des formes 3D simples ou VQA (Antol et al. 2015) qui porte sur des photos. Quelques exemples de questions dans CLEVR sont présentés en figure 5.

2.1.2. Introduction du sujet

Dans le travail qui va suivre, je m'intéresse à deux jeux de données, OpenBookQA (Mihaylov et al. 2018) et ARC (P. Clark, Cowhey et al. 2018), représentatifs des bases de données de questions à choix multiples. OpenBookQA et ARC comportent des questions associées à 4 réponses possibles similairement à la base de données RACE, mais contrairement à cette dernière, ne fournissent pas de texte de référence avec chaque question.



Are there an equal number of large things and metal spheres ?

What size is the cylinder that is left of the brown metal thing that is left of the big sphere ?

There is a sphere with the same size as the metal cube ; is it made of the same material as the small red sphere ?

How many objects are either small cylinders or metal things ?

Figure 5 – Un exemple de question dans CLEVR

Au lieu de cela, les deux bases de données sont associées à un ensemble de connaissance générale sous la forme de phrases courtes écrites en langage naturel (par exemple : “A bee is a pollinating animal.”) censées contenir toutes les connaissances nécessaires pour répondre aux questions mais qui ne sont pas directement liées à une question en particulier. Les modèles peuvent ainsi extraire de l'information de cet ensemble de phrases afin de répondre à la question proposée. Cependant, au moment de réaliser les expériences décrites dans ce chapitre, la plupart des modèles état de l'art choisissaient encore d'ignorer ces informations supplémentaires et s'appuyaient plutôt sur une connaissance du monde apprise pendant leur entraînement. Et pourtant les intérêts d'un tel modèle sont multiples : d'un côté, l'utilisation de données externes plutôt que de se reposer sur une connaissance apprise permet en théorie de réduire la taille des modèles. En effet, les modèles n'ont alors pas besoin de stocker l'intégralité de la connaissance du monde pour pouvoir répondre à une question et peuvent se concentrer sur la logique d'extraction d'information. De plus, un tel modèle capable d'extraire de l'information depuis une base de connaissance, peut plus facilement être converti pour être utilisé sur de nouvelles bases de données en ajoutant ou

en remplaçant de l'information dans la base de connaissances et sans avoir à entraîner le modèle à nouveau. Il faut noter que cette tendance a largement évolué dans les dernières années et de plus en plus de modèles ont depuis utilisé cette information externe que ce soit sur OpenBookQA ou ARC en utilisant les bases de données de connaissances générales fournies avec ceux-ci ou sur d'autres datasets en utilisant des sources de connaissance générale externes comme Wikipédia.

Cependant, mes expériences préliminaires ont montré que l'ajout d'une telle information supplémentaire (via par exemple un mécanisme d'attention) ne permet pas d'obtenir d'amélioration significative des résultats des modèles. En tentant de comprendre pourquoi l'ajout d'information dans mes architectures ne menait pas à des gains, nous avons inspecté les modèles et avons observé que ces derniers répondent sans nécessairement faire bon usage de la question, en exploitant des biais dans les jeux de données. Dans la suite de ce chapitre, nous rapportons donc les résultats de plusieurs expériences que nous avons menées avec des modèles de type BERT sur OpenBookQA et ARC. Ces expériences mettent en évidence des biais présents dans les jeux de données utilisées et nous tentons par la suite d'éliminer ces biais grâce à différentes méthodes de régularisation de l'apprentissage des modèles.

2.2. Etat-de-l'art

En 2019, Devlin et al. (2019) introduisent BERT, un modèle d'apprentissage profond pré-entraîné qui a montré d'énormes améliorations dans un grand nombre de tâches de TAL, y compris le question-réponse. Les modèles inspirés de BERT ainsi que d'autres modèles pré-entraînés basés entre autres sur T5 (Raffel et al. 2020) ou GPT (Radford, J. Wu et al. 2019)(Brown et al. 2020) sont actuellement largement utilisés sur de nombreuses bases de données de question-réponse et occupent souvent les premières places dans les classements. Ces modèles battent même les performances humaines sur certains datasets comme c'est le cas sur SQuAD (SQuAD Leaderboard).

Ces modèles pré-entraînés peuvent tirer parti d'une quantité massive de données non étiquetées et sont donc particulièrement utiles pour les tâches qui nécessitent une bonne connaissance générale du monde puisqu'ils intègrent déjà certaines connaissances sémantiques issues de leur pré-entraînement. De plus, le pré-entraînement permet de réduire le temps d'entraînement sur des données spécifiques et la quantité de données nécessaire. Ce

Modèles	Accuracy (%)
Annotateurs Humains	91.7
X-Reasoner	94.2
GenMC (ensemble)	92.0
GenMC (Huang et al. 2022)	89.8
PipeNet	88.2
DRAGON	87.8

Tableau 1 – Accuracy des meilleurs modèles de l'état de l'art pour OpenBookQA. Source : OpenBookQA Leaderboard (Septembre 2022).

Modèles	Accuracy (%)
ST-MoE-32B	86.5
UnifiedQA + ARC MC/DA + IR (Tafjord et P. Clark 2021)	81.4
UnifiedQA-v2(T5-11B) (Khashabi, Kordi et Hajishirzi 2022)	81.1
GenMC (Huang et al. 2022)	79.9
ZeroQA	78.6

Tableau 2 – Précision des meilleurs modèles de l'état de l'art pour ARC Challenge. Source : ARC Leaderboard (Septembre 2022).

pré-entraînement est donc particulièrement avantageux pour des ensembles de données relativement petits comme ARC et OpenBookQA qui ne rassemblent chacun que quelques milliers de questions. Presque tous les modèles état de l'art actuels sur ARC et OpenBookQA sont des modèles pré-entraînés. Les tableaux 1 et 2 montrent les meilleurs modèles actuels sur OpenBookQA et ARC respectivement.

À la suite de la publication de l'article initial, de multiples adaptations de BERT ont été proposées – XLNet (Yang, Dai et al. 2019), RoBERTa (Y. Liu et al. 2019), etc – chacune améliorant le modèle ou la procédure de pré-entraînement. Sentence-BERT (SBERT) (Reimers et Gurevych 2019) est l'un de ces modèles qui visent à augmenter la significativité sémantique de la représentation de phrase fournie par BERT. Les auteurs proposent d'ajouter au préapprentissage BERT standard une étape de pré-entraînement supplémentaire sur le jeu de données SNLI (Bowman et al. 2015). SNLI est un ensemble de données contenant des paires de phrases étiquetées comme implication, contradiction ou neutre. Ils affinent

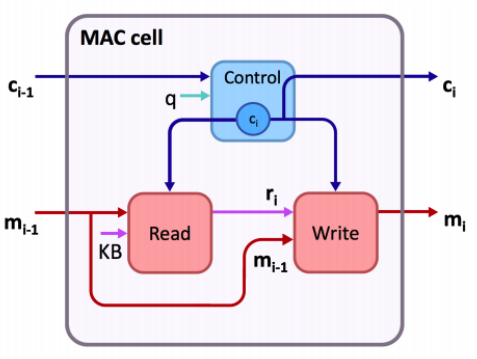


Figure 6 – Fonctionnement interne d'une "MAC Cell". La cellule est une unité récurrente maintenant 2 vecteurs d'état c_i et m_i. c_i est l'état de contrôle et détermine quelle action doit être effectuée à un pas de temps donné et m_i est une mémoire qui stocke les informations des étapes précédentes. Image de Hudson et Manning 2018

alors BERT en utilisant une architecture siamoise afin que deux phrases marquées comme implication aient une représentation proche l'une de l'autre (distance cosinus) et que deux phrases marquées comme contradiction aient des représentations éloignées. Leur proposition est que les représentations de phrases résultantes sont sémantiquement plus pertinentes que les représentations obtenues par un modèle BERT standard.

Dans les expériences qui suivent, j'utilise également les "MAC Cells" (Hudson et Manning 2018). Cette architecture a d'abord été introduite sur le jeu de données CLEVR (Johnson et al. 2017) (question-réponse utilisant une image) afin d'améliorer la capacité de raisonnement des réseaux de neurones basés sur un mécanisme d'attention. Ce modèle est construit comme un réseau récurrent maintenant deux vecteurs d'état (mémoire et contrôle). Le vecteur de contrôle est utilisé pour déterminer quelle action de raisonnement doit être effectuée à chaque étape (lire de l'information dans une base de données ou bien modifier l'état de la mémoire) tandis que le vecteur mémoire est une représentation de toutes les informations que le modèle a obtenues jusque-là. La cellule MAC est composée de 3 modules (voir figure 6) : à chaque étape le module "contrôle" met à jour le vecteur de contrôle en utilisant le vecteur de contrôle précédent. Un module de "lecture" utilise alors ce nouveau vecteur de contrôle et le vecteur mémoire précédent pour effectuer une attention sur une base de données (attention sur l'image dans le cas de CLEVR) et crée ainsi une proposition de nouveau vecteur mémoire. La nouvelle mémoire finale est une combinaison linéaire de la mémoire précédente et de la nouvelle proposition. La proportion d'ancienne et de nouvelle information est décidée par le module "écriture" en fonction de l'état de la commande. Les

"MAC Cells" ont obtenu de bons résultats sur CLEVR et ont montré qu'elles avaient une bonne capacité à raisonner en combinant différents éléments de la base de données.

2.3. Protocole expérimental

Pour cette étude, j'utilise deux jeux de données : OpenBookQA (Mihaylov et al. 2018) et ARC (P. Clark, Cowhey et al. 2018). Les deux sont des bases de données de questions à choix multiples avec 4 choix de réponse pour chaque question. Les questions portent sur une grande variété de sujets liés à la logique de la vie quotidienne ou aux connaissances générales.

Le jeu de données OpenBookQA est composé de 3 parties : entraînement, validation et test avec respectivement 4958, 500 et 500 questions chacune. OpenBookQA est accompagné de deux petits ensembles de données de phrases de connaissances générales qui sont similaires dans leur contenu. Cependant l'un d'eux est composé de toutes les phrases fournies aux annotateurs en guise d'inspiration lors de la création du jeu de données (chaque question est liée à une phrase mais chaque phrase peut avoir été utilisée pour plusieurs questions) et par conséquent ces phrases contiennent la plupart du temps les informations nécessaires pour répondre aux questions. La correspondance entre les questions et ces phrases est connue. Le second ensemble est composé du même genre de phrases mais celles-ci ne sont directement liées à aucune question en particulier. Le premier et deuxième ensemble de phrases sont composés respectivement de 1327 et 5168 phrases. Voir la figure 7 pour des exemples de questions et de phrases associées.

La métrique utilisée pour l'évaluation est l'*accuracy* définie comme le pourcentage de questions correctement répondues. Le meilleur modèle actuel sur OpenBookQA est *X-Reasoner* avec un score d'accuracy de 94% (OpenBookQA Leaderboard). *X-Reasoner* est constitué d'un extracteur d'information (basé sur *RocketQA* (Qu et al. 2021) et SBERT (Reimers et Gurevych 2019)) chargé de sélectionner des phrases dans la base de données de faits scientifiques, et d'un modèle T5-11B chargé d'utiliser ces faits pour sélectionner la bonne réponse. BERT Large, quant à lui, atteint une accuracy de 60% sans pré-entraînement supplémentaire. Sur ARC, le meilleur modèle, *ST-MoE-32B*, obtient une accuracy de 87% (ARC Leaderboard) d'accuracy sur l'ensemble "*Challenge*" alors que les

OpenBookQA	
Question	Stars are ... - A : warm lights that float B : made out of nitrate **C : great balls of gas burning billions of miles away** D : lights in the sky
Related fact	A star is made of gases
Other interesting facts	▷The Earth rotating on its axis causes the sun to appear to move across the sky at night ▷The Earth rotating on its axis causes stars to appear to move across the sky at night ▷The north star does not move in the sky in the Northern ▷Hemisphere each night ▷Burning wood is used to produce heat

ARC	
Question	Which is a nonrenewable resource ? - **A : oil** B : trees C : solar energy D : food crops

Figure 7 – Un exemple de 2 questions dans ARC et OpenBookQA avec le fait donné à l'annotateur et 4 faits supplémentaires sélectionnés automatiquement par co-occurrence de mots entre la question et tous les faits possibles.

scores de BERT Base et Large sont respectivement d'environ 36% et 40% [1]. Au moment de la réalisation des expériences décrites ci-après, les meilleurs modèles disponibles étaient *AristoRoBERTaV7* (P. Clark, Etzioni et al. 2020) sur OpenBookQA avec un score d'accuracy de 78% et *FreeLB-RoBERTa* (C. Zhu et al. 2020) sur ARC Challenge avec une accuracy de 68%.

1. les scores de BERT ne sont pas disponibles sur le leaderboard d'ARC et sont donc basés sur mon implémentation.

2.4. Modèles

Dans cette section, je présente les différentes architectures de modèles que j'utilise dans cette étude. Dans tous ces modèles, j'utilise BERT ou SBERT pour obtenir des représentations vectorielles de phrases. Dans les descriptions et schémas qui vont suivre, j'utiliserai (S)BERT pour décrire le fait que BERT et SBERT sont interchangeables dans ces configurations. De plus, bien que je ne les utilise pas ici, ces configurations seraient aussi applicables à tout autre modèle pré-entraîné de type BERT comme RoBERTa ou ALBERT.

J'utilise par la suite les notations suivantes : pour une question q, nous nommons $(r_i)_{i\in\{1,2,3,4\}}$ les différents choix de réponse possibles pour la question q et $(c_j)_{j\in\{1,\dots,n\}}$ un ensemble de n phrases de connaissance générale associé à la question q. Dans les cas où une seule phrase de connaissance (phrase d'aide) est fournie, celle-ci est simplement nommée c. Le choix de réponse correct est quant à lui dénoté par r_{i^*}.

Pour le question-réponse, lorsqu'un modèle de type BERT est utilisé, l'entraînement des modèles se fait comme suit (La méthode d'entraînement diffère lorsque que l'on utilise des modèles séquence à séquence comme T5) : un score scalaire $s(q, r_i)$ est d'abord calculé en utilisant (S)BERT pour chaque couple question + choix de réponse possible. Ces scores sont ensuite passés à une fonction softmax pour obtenir une distribution de probabilité :

$$P_i = P(r_i|q) = \frac{s(q,r_i)}{\sum_j s(q,r_j)}$$

et l'entraînement est réalisé à l'aide d'une perte L basée sur une entropie croisée :

$$L = H(P^*, P) = -\sum_i P^*(r_i|q) log\, P(r_i|q) = -log\, P(r_{i^*}|q)$$

où P^* désigne la distribution réelle des bonnes réponses :

$$P^*(r_i|q) = \begin{cases} 1 \text{ si } i = i^* \\ 0 \text{ sinon} \end{cases}$$

Au moment de l'inférence, la réponse qui obtient le meilleur score est sélectionnée par le modèle. Pour mes expériences, je teste différentes configurations pour le calcul des scores $s(q, r_i)$ qui sont décrites ci-après.

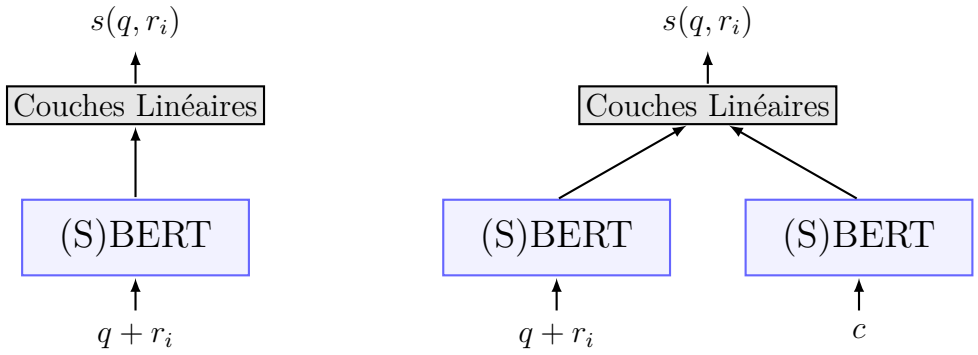
2.4.1. Configuration A

Cette configuration schématisée en figure 8(a) est la méthode standard d'utilisation de (S)BERT pour le question-réponse utilisé par la grande majorité des modèles proposés sur ARC et OpenBookQA. Les choix de réponses sont concaténés à la question obtenant ainsi 4 séquences $(q + r_i)_{i\in\{1,2,3,4\}}$ (où + désigne la concaténation de deux séquences de mots). À partir de là, nous utilisons (S)BERT pour obtenir un vecteur de représentation pour chaque choix et nous envoyons ce vecteur de représentation à travers un perceptron à 2 couches avec une fonction d'activation ReLU pour obtenir le score scalaire pour chaque choix de réponse. Il existe plusieurs méthodes pour obtenir la représentation vectorielle d'une phrase avec (S)BERT : BERT fourni une représentation vectorielle pour chaque token dans la phrase et il est alors possible de soit moyenner toutes ces représentations ou bien simplement prendre la représentation correspondant au premier token dans la phrase. Du moment que (S)BERT est affiné par la suite sur la tâche, je n'ai pas observé de différences significatives entre les deux méthodes et j'utilise donc une moyenne.

J'utiliserai aussi par la suite une version de cette configuration où une phrase d'aide c est concaténée à la séquence d'entrée de (S)BERT obtenant ainsi des séquences d'entrée de la forme $(c + q + r_i)_{i\in\{1,2,3,4\}}$.

2.4.2. Configuration B

Dans cette architecture visualisée en figure 8(b), en plus de la représentation de la séquence $q + r_i$, je fournis au modèle une phrase supplémentaire c dont la représentation vectorielle est également calculée avec (S)BERT. Il convient de noter que ce modèle et les



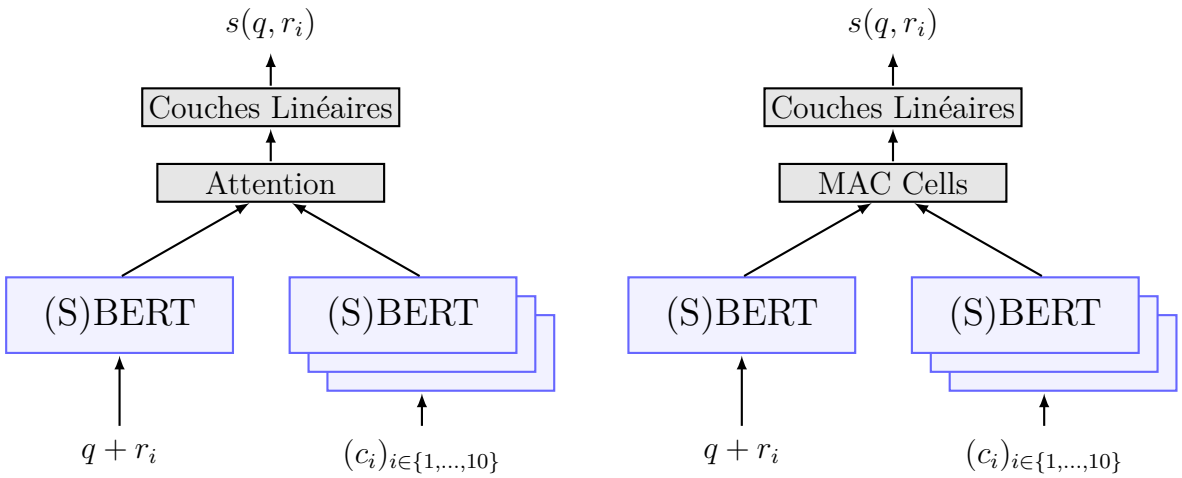
(a) Configuration A **(b)** Configuration B

Figure 8 – Description du modèle A (à gauche) et B (à droite). Un score scalaire est calculé à partir de chaque choix de réponse. Les scores pour tous les choix de réponse sont ensuite rassemblés et passés par une fonction softmax (omise ici).

suivants sont entraînés sur uniquement OpenBookQA car ils nécessitent que chaque question soit liée à une phrase de connaissance générale servant de phrase d'aide. J'utilise la phrase de connaissance générale associée à la question dans OpenBookQA. Cette phrase était fournie aux annotateurs comme source d'inspiration pour écrire la question et donc, est censée donner suffisamment d'informations au modèle pour répondre à la question. L'idée derrière ce modèle est d'évaluer la qualité sémantique des représentations de phrase fournies par BERT. Étant donné que la phrase supplémentaire est censée avoir un sens plus proche de la réponse que des autres choix, leurs représentations devrait également être plus proches. La représentation de la phrase d'aide est concaténée à la représentation de la séquence $q + r_i$ et transmise au perceptron final pour obtenir le score $s(q, r_i)$.

2.4.3. Configuration C

Le modèle B n'est pas applicable à un cas réel puisqu'il suppose d'avoir accès à une seule phrase qui donne suffisamment d'information pour répondre à la question. Dans cette troisième architecture schématisée en figure 9(a), j'ajoute un mécanisme d'attention à la Configuration B. Au lieu d'une phrase unique, le modèle doit maintenant sélectionner la bonne phrase parmi un ensemble de 10 phrases $(c_i)_{i\in\{1,\dots,10\}}$ extraites du jeu de données de connaissances générales d'OpenBookQA. Je sélectionne pour cela les 9 phrases avec le plus grand nombre de mots en commun avec la question et j'ajoute la phrase "cible" si elle n'est pas déjà sélectionnée.



(a) Configuration C **(b)** Configuration D

Figure 9 – Description du modèle C (à gauche) et D (à droite). La seule différence avec le modèle B est qu'au lieu d'une seule phrase d'aide, le modèle a accès à plusieurs phrases incluant l'aide et doit sélectionner la bonne avec un mécanisme d'attention (modèle C) ou des cellules MAC (modèle D).

2.4.4. Configuration D

Enfin, j'ai expérimenté avec les "MAC Cells" (Hudson et Manning 2018). Habituellement, plusieurs faits peuvent être pertinents pour une question donnée. La capacité de raisonnement des MAC Cells peut ainsi être utile pour assembler de multiples informations à partir de plusieurs faits. Nous remplaçons la simple attention du modèle C par une cellule MAC dans l'espoir que cela aide le modèle à extraire plus d'informations depuis plusieurs phrases. Voir la figure 9(b).

2.5. Détails d'implémentation

J'utilise l'implémentation Pytorch de BERT proposé par Hugging Face (Wolf et al. 2020) et l'implémentation de SBERT fournie avec l'article original [2]. J'utilise la version *base* de BERT et SBERT. Cela signifie que les représentations de mots sont de dimension 768 et j'ai gardé la même taille de vecteur de représentation pour les éventuels mécanismes d'attention et couches linéaires des différentes configurations. J'ai implémenté le reste du code en utilisant Pytorch.

2. https://www.sbert.net/

L'entraînement de tous les modèles se fait avec des mini-batchs de taille 8. Les phrases les plus courtes du mini-batch sont rembourrées à l'aide d'un token spécial et je tronque les séquences de plus de 40 tokens en supprimant les premiers tokens (pas les derniers) pour s'assurer de garder le choix de réponse intact (le tableau 5 fournit des statistiques de longueur pour OpenBookQA). j'entraîne les modèles pour un maximum de 5 époques, après quoi les modèles commencent à sur-apprendre car les ensembles de données sont petits et (S)BERT a une très grande capacité. Pour compenser le sur-apprentissage, les poids de (S)BERT sont gelés pendant les 2 premières époques pour laisser une chance aux autres parties du réseau de converger. Sur les modèles nécessitant un mécanisme d'attention, j'ajoute une fonction de coût supplémentaire (entropie croisée) directement sur la distribution de l'attention pour aider les modèles à identifier plus rapidement quelle est la phrase qui donne l'information requise. Un coefficient de pondération est appliqué à cette partie de la perte. Il commence à 1,0 et diminue à 0,2 après les 2 premières époques. Tous les hyperparamètres ci-dessus sont choisis pour maximiser l'*accuracy* sur l'ensemble de validation.

Lors de l'entraînement sur le jeu de données ARC, j'utilise à la fois les données d'entraînement des parties "Easy" et "Challenge" mais je rapporte des résultats séparés pour le test.

2.6. Résultats

Dans cette section, je présente les expériences que j'ai réalisées avec BERT et SBERT sur OpenBookQA et ARC. Toutes les expériences présentées ci-dessous ont donc été réalisées dans le but d'expliquer pourquoi, dans mes expériences préliminaires, je n'avais pas réussi à appliquer avec succès un mécanisme d'attention (modèle C) sur une base de connaissances de phrases. En effet, l'objectif initial de mes expériences était d'appliquer la configuration C et de permettre au modèle, au moment d'évaluer la justesse d'un choix de réponse, de venir piocher de l'information dans un ensemble de phrases (dont j'obtiens la représentation vectorielle par (S)BERT) grâce à un mécanisme d'attention. Ceci suppose que les représentations proposées par (S)BERT incorporent suffisamment de la sémantique des phrases et des questions pour que le mécanisme d'attention soit capable de déterminer quelles phrases sont les plus intéressantes pour répondre à la question considérée. En pratique, cela ne semble pas être le cas et j'ai cherché à comprendre pourquoi.

	question entière	4 tokens	pas de question
BERT (model A)	**55.8**	52.0	51.2
SBERT (model A)	53.2	**53.0**	**53.6**
BERT (model A + help)	**64.5**	64.9	-
SBERT (model A + help)	63.6	**65.2**	-
BERT (model B)	53.0	53.4	-
SBERT (model B)	**55.0**	**61.3**	-
SBERT (model C)	54.8	56.8	-
SBERT (model D)	**56.8**	**56.8**	-

Tableau 3 – Récapitulatif des scores d'*accuracy* obtenus (en pourcentage de réponses correctes) sur l'ensemble de test d'OpenBookQA selon que l'on utilise un modèle standard, un modèle où on donne une phrase d'aide (phrase ayant servi d'inspiration pour la question) en parallèle ou bien un modèle avec un mécanisme d'attention sur un ensemble de faits (comprenant la phrase d'aide). Ce tableau compare aussi les résultats des modèles selon qu'on utilise la question entière, les 4 derniers tokens ou pas de question du tout.

2.6.1. Découverte et analyse des biais

Tout d'abord, afin d'évaluer quelles parties d'une question BERT utilise pour y répondre, je diminue le nombre de tokens de la question que je donne aux modèles pendant l'entraînement et le test. L'idée est que si on retire le début d'une question et que le modèle n'a accès qu'aux quelques derniers mots de celle-ci, nous devrions observer une diminution importante de ces performances. Pour ce faire, j'utilise la configuration A décrite dans la section 2.4. Je donne les résultats (en termes d'*accuracy*) d'un modèle qui a accès à la question complète et d'un modèle qui n'a accès qu'aux 4 derniers tokens de la question. Finalement, je supprime entièrement la question, donnant ainsi uniquement le choix de réponse a BERT. Je m'attendais bien sûr à ce que le modèle atteigne une *accuracy* d'environ 25% (puisque OpenBookQA a 4 choix de réponses pour chaque question) avec cette configuration puisque sans la question, il n'y a supposément aucun moyen de différencier la bonne réponse des autres choix. J'ai également vérifié que les jeux de données (train, validation et test) sont équilibrés dans le sens où tous les choix de réponses (A, B, C et D) apparaissent dans environ 25% des réponses chacun.

	question entière	4 tokens	pas de question
BERT (partie "*Easy*")	51.7	46.9	36.5
BERT (partie "*Challenge*")	36.5	36.2	34.2

Tableau 4 – Récapitulatif des scores d'*accuracy* (en pourcentage de réponses correctes) obtenus sur les ensembles de test d'ARC "*Easy*" et "*Challenge*".

longueur en mots (entraînement/test)	moyenne	min	max
Questions	10.7/10.3	1/1	68/61
Mauvaises réponses	2.7/3.0	1/1	20/16
Bonnes réponses	3.0/3.3	1/1	21/15

Tableau 5 – Longueurs moyenne/minimum/maximum des questions et des réponses dans OpenBookQA. Les bonnes réponses ont tendance à être légèrement plus longues que les mauvaises réponses.

Le tableau 3 montre les scores d'*accuracy* résultants sur les ensembles de validation et de test d'OpenBookQA (2 premières lignes) pour BERT et SBERT (à des fins de comparaison puisque nous utilisons SBERT dans des expériences ultérieures). Ce qui ressort de cette première expérience est que, étonnamment, ne donner que les 4 derniers tokens de la question à BERT ne diminue que légèrement les performances. De plus, si je supprime complètement la question, BERT parvient toujours à obtenir une *accuracy* supérieure à 50%, pas si éloignée de l'*accuracy* d'origine et même, dans le cas de SBERT, une meilleure *accuracy*.

J'ai mené la même expérience sur les deux parties (*Easy* et *Challenge*) d'ARC. Les résultats sont rapportés dans le tableau 4. Les résultats de l'ensemble *Challenge* sont assez similaires à ce que nous avons observé pour OpenBookQA : l'*accuracy* pour 4 tokens et pas de question est proche de l'*accuracy* obtenue par BERT dans une configuration standard avec toute la question. Sur l'ensemble *Easy* cependant, nous pouvons constater une nette dégradation de l'*accuracy* lors de la suppression des tokens mais nous obtenons toujours 36% dans le pire des cas, ce qui est nettement meilleur que les 25% attendus pour un modèle aléatoire. Cela montre que dans tous les cas, il est possible de répondre à une partie importante des questions sans même regarder la question elle-même.

Fréquence des mots (entraînement/test)	moyenne	min	max
Mauvaises réponses	0.0054/0.0056	0.00029/0.00033	0.016/0.017
Bonnes réponses	0.0050/0.0058	0.00014/0.00019	0.016/0.019

Tableau 6 – Ce tableau montre les fréquences moyennes des mots les moins fréquents et des mots les plus fréquents dans les bonnes et mauvaises réponse ainsi que la moyenne des fréquences des mots qui apparaissent dans celles-ci. On remarque qu'en moyenne, les bonnes réponses contiennent des mots qui sont moins fréquents et donc plus spécifiques.

Mes modèles apprennent donc en partie quelles sont les caractéristiques d'une bonne réponse au lieu d'apprendre le lien logique entre une question et la réponse correspondante. J'ai trouvé quelques éléments d'explication à cela. Tout d'abord, je calcule des statistiques sur la longueur des questions et des réponses. Ces statistiques sont rapportées dans le tableau 5 et montrent que les bonnes réponses sont en moyenne plus longues que leurs homologues. En pratique, un modèle factice qui sélectionne les réponses les plus longues parmi les 4 proposées atteint une *accuracy* de 33% ce qui n'explique donc pas l'intégralité des 51,2% obtenus par BERT sur OpenBookQA.

Dans le tableau 6, je fournis en outre des statistiques sur les fréquences relatives des tokens dans les réponses correctes et incorrectes. Ce qui est intéressant ici, c'est que le token le moins fréquent dans les bonnes réponses est en moyenne moins fréquent que le token le moins fréquent dans les mauvaises réponses. Cela signifie que les réponses correctes ont tendance à inclure des mots plus spécifiques que les réponses incorrectes. Cela pourrait être un biais lié à la méthode d'annotation dans laquelle les annotateurs sont obligés d'inventer des réponses incorrectes. Il est possible qu'un annotateur ait tendance à être plus générique lorsqu'il essaie d'écrire une mauvaise réponse sans inspiration. Un modèle factice qui sélectionne la réponse avec le token le moins fréquent atteint une *accuracy* de 36,8% sur OpenBookQA. Enfin, nos expériences montrent que 54,6% des réponses correctes sont soit les séquences les plus longues, soit celle contenant le jeton le moins fréquent. Ces deux expériences bien qu'elles n'expliquent pas l'intégralité du phénomène, montrent que les modèles (BERT et SBERT) peuvent être fortement biaisés par des facteurs étrangers à la logique du question-réponse.

Question	What impacts an objects ability to reflect light ?
Answer choices	**A : color pallete** B : weights C : height D : smell
4 tokens	...ability to reflect light ?

Figure 10 – Un example de questions dans OpenBookQA (l'erreur d'orthographe est présente dans le dataset) pour laquelle il est relativement facile de déterminer la réponse sans lire la question. La réponse correcte est plus longue et contient des mots plus complexes que les autres.

2.6.2. Ajout de connaissance avec un oracle

Comme expliqué dans la section 2.3, OpenBookQA fournit avec chaque question une courte phrase de connaissance générale qui a été donnée à l'annotateur comme source d'inspiration pour la question. Bien que l'utilisation directe de cette phrase rende les résultats incomparables avec les résultats présentés ci-dessus et ceux de l'état de l'art, nous pouvons l'utiliser pour évaluer l'*accuracy* qui pourrait être obtenue si nous pouvions faire une sélection correcte de connaissances connexes dans une base de données. J'appelle cette phrase particulière la phrase d'aide dans cette section.

Dans ce qui suit, j'utilise toujours (S)BERT dans la configuration A sur OpenBookQA mais au lieu de la simple séquence question + réponse, je concatène au début de cette séquence la phrase d'aide liée à la question. L'idée pour l'instant est de voir si les représentations issues de (S)BERT sont utilisables dans une configuration où la solution est directement donnée au modèle. Le tableau 3 montre que, comme prévu, nous obtenons une augmentation significative des performances.

Maintenant, que se passe-t-il si la nouvelle phrase n'est pas donnée avant le passage par (S)BERT mais plutôt après celui-ci ? Pour tester cela, nous utilisons la configuration B. Dans cette configuration, le modèle a donc besoin d'obtenir une représentation de phrase suffisamment significative de la part de (S)BERT pour répondre puisqu'il doit évaluer la similarité entre la phrase d'aide et chacun des choix de réponse. La précision est à nouveau décrite dans le tableau 3 (lignes 5 & 6). Nous observons que BERT fonctionne relativement

mal dans cette tâche et ne montre aucun changement significatif de sa précision entre cette configuration et aucune phrase d'aide. SBERT obtient cependant un gain similaire à ce qui se passe lors de la concaténation de la phrase d'aide à la question. Ainsi, les représentations de SBERT semblent plus adaptées pour trouver des liens de similarité sémantique entre phrases.

Pour confirmer cette observation, j'ai exécuté un autre modèle simple. Ce modèle calcule la similarité (distance cosinus) entre la représentation donnée par (S)BERT de la phrase d'aide et les représentations des questions + réponses. Le modèle sélectionne ensuite le choix de réponse avec la plus grande similarité. Le processus se fait sans aucun entraînement sur OpenBookQA en utilisant uniquement les BERT et SBERT pré-entraînés. En utilisant ce modèle, SBERT atteint une précision de 57,6% sur l'ensemble de test d'OpenBookQA, tandis que BERT n'atteint que 37,4%, démontrant une fois de plus que les représentations produites par SBERT contiennent plus de la sémantique des phrases et sont donc plus adaptées à la tâche.

2.6.3. Ajout de connaissance avec un scénario plus réaliste

Enfin, j'ai testé des configurations plus réalistes. Au lieu de donner directement la phrase d'aide, je la "cache" parmi d'autres phrases similaires, simulant ainsi un scénario dans lequel je dispose d'un algorithme capable de sélectionner avec précision un ensemble de phrases factuelles utiles pour répondre à la question. Comme expliqué dans la section 2.4, je sélectionne 9 phrases dans l'ensemble de données factuelles d'OpenBookQA en fonction du nombre de mots en commun maximal avec la question et j'ajoute la phrase d'aide pour assurer une bonne sélection. Je reporte les résultats de SBERT pour les configurations C et D dans le tableau 3. BERT fonctionne mal pour les deux configurations (probablement pour les raisons exposées précédemment) et donc seule la précision de SBERT est indiquée ici. Dans l'ensemble, les modèles C et D sont plus faibles que la configuration B avec aide directe mais fonctionnent toujours mieux que le modèle A sans aide. Cela tend à montrer qu'il y a de la valeur à gagner en ajoutant des informations supplémentaires de culture générale aux entrées du modèle. Une manière de débiaiser les datasets pourrait être de choisir les distracteurs pour une question parmi les bonnes réponses à d'autres questions. Ainsi, nous pourrions potentiellement améliorer l'entraînement de BERT en supprimant le minimum local créé par le biais de longueur et de fréquences.

2.7. Première approche : réduire l'apprentissage des biais

La première approche que j'ai étudiée a été la réduction de l'apprentissage des biais qui existent dans les données lors de l'entraînement des modèles. Les méthodes de réduction de biais existantes se séparent en deux approches principales :

— L'utilisation de plusieurs bases de données en même temps lors de l'entraînement. L'idée est qu'en amenant des données venant de différentes annotations, différents domaines, etc., les biais qui peuvent exister dans certaines bases de données ne seront pas présents dans d'autres et inversement. Les modèles apprennent donc seulement les *features* qui apparaissent dans un grand nombre de bases de données et qui sont donc a priori pertinentes pour la tâche. Les limitations de ce type de méthodes, sont que d'une part certains biais peuvent persister si par exemple plusieurs bases de données ont été annotées de manière similaire et donc qu'elles contiennent les mêmes types de biais. De plus, ceci n'améliore pas vraiment le manque de données annotées qui existe dans certaines tâches de question-réponse d'autant que, et cela rejoint le premier point, il faut trouver des données annotées de manière indépendante pour limiter le risque qu'un biais se retrouve dans plusieurs ensembles.

— L'utilisation de méthodes adversarielles, comme Kim et al. (2019) par exemple. L'idée est la suivante : si l'on dispose sur un ensemble d'entraînement d'un biais connu, alors on peut ajouter au niveau d'une des représentations intermédiaires du modèle (comme par exemple entre l'encodeur et le décodeur dans un réseau encodeur-décodeur) un sous-réseau ayant pour rôle d'essayer de prédire ce biais. On peut alors, en parallèle de l'erreur de reconstruction ou de classification classique du réseau, utiliser une erreur adversarielle qui entraîne le réseau encodeur à pénaliser le sous-réseau de prédiction du biais et donc à ne pas transmettre d'information sur ce biais ou pouvant permettre de retrouver ce biais. Privé de cette information, le décodeur n'a donc pas d'autre choix que de trouver d'autres manières de résoudre sa tâche en utilisant de l'information plus pertinente. Adeli et al. (2020) est un autre exemple d'une telle approche. Les limitations de cette approche est qu'elle présuppose que l'on sait quel type de biais est présent dans la base de données, ce qui implique un travail d'investigation préalable

sur la tâche. Si on prend l'exemple du question-réponse, nous avons déterminé dans (Le Berre et Langlais 2020) que la longueur des phrases et la fréquence des mots utilisés sont des biais importants dans OpenBookQA, mais il est difficile de dire si ce sont ces biais qui sont utilisés par les modèles et de plus ces biais à eux seuls ne permettent pas d'expliquer les résultats des modèles sur cette tâche et il en existe donc probablement d'autres.

Je me suis donc tourné vers deux autres approches. Dans un premier temps j'ai étudié la possibilité d'utiliser des méthodes de régularisation pour pénaliser les modèles qui n'apprennent pas de l'information pertinente. J'ai également étudié une méthode adversarielle.

2.7.1. Régularisateurs

Cette section présente différentes méthodes de régularisation de l'apprentissage des modèles de question-réponse. L'idée est d'ajouter en parallèle de l'erreur classique (entropie croisée entre les différents choix de réponse dans le cas des questions à choix multiples) une erreur qui va pénaliser l'apprentissage du biais ou qui au contraire va pousser le modèle à apprendre de l'information intéressante.

J'ai pour cette étude travaillé avec OpenBookQA et ARC, les deux bases de données pour lesquelles j'ai la certitude qu'il existe des biais qui faussent les résultats des modèles d'apprentissage profond. Les baselines utilisées sont BERT et RoBERTa dans leurs versions "Base" et "Large" et en utilisant la configuration classique de ces modèles pour les questions à choix multiples (modèle pré-entraîné affiné avec une couche linéaire sur le vecteur de représentation du premier token).

Les méthodes de régularisation présentées ci-après résultent de l'observation suivante : lors de l'apprentissage d'un modèle de type BERT sur une tâche de question-réponse à choix multiples, BERT, RoBERTa, ALBERT, etc. produisent un vecteur de représentation pour chaque choix de réponse et une couche linéaire transforme ce vecteur en un score scalaire. Ce processus est décrit plus en détail dans la section 2.4. On utilise ensuite un softmax et une perte d'entropie croisée pour l'apprentissage du modèle. Intuitivement, on s'attend à ce que les scores correspondant aux bonnes réponses soient clairement et significativement distincts de ceux correspondant a des mauvaises réponses. On devrait donc, si on affiche la distribution des scores obtenues par tous les exemples, positifs ou négatifs, obtenir 2 clusters de scores :

l'un correspondant aux réponses correctes (∼25% des exemples dans le cas d'OpenBookQA) et l'autre aux distracteurs (∼75% des exemples). En effet, en théorie, le modèle attribue à chaque couple question+réponse un score en fonction de la probabilité qu'a ce couple d'être correct et un couple correct est toujours correct même quand on le compare avec d'autres couples distracteurs issus d'autres questions. Or, en pratique les scores obtenus sont tous regroupés en une seule et même gaussienne centrée sur zéro. De plus, un couple question-réponse correct et classifié comme correct par le modèle lorsque comparé aux distracteurs pour la même question peut se retrouver avoir un score très faible par rapport à d'autres réponses fausses dans la base de données. Une explication probable à ceci est la méthode d'entraînement utilisée, basée sur une fonction softmax et une entropie croisée. En effet, avec une telle méthode, on n'entraîne pas le modèle à déterminer ce qui fait d'un choix de réponse une bonne réponse mais on l'entraîne plutôt à classer différents choix de réponse en fonction de leur vraisemblance ou de leur relation avec la question. Un corollaire direct de ce fait est que le modèle n'a pas besoin de donner un score particulièrement haut à une bonne réponse et doit juste s'assurer que le score obtenu par la bonne réponse soit plus grand que les scores obtenus par les autres choix de réponse pour la même question indépendamment des autres questions dans la base de données. Or on aimerait que le score donné par le modèle pour une réponse soit un bon indicateur de la probabilité de cette réponse d'être correcte indépendamment des distracteurs associés.

L'une des premières idées a donc été de rajouter des erreurs de régularisation permettant de forcer les modèles à rassembler les réponses correctes d'un côté et les réponses fausses de l'autre afin de pousser ces modèles à se concentrer sur les traits qui permettent de discriminer de manière fiable les bonnes des mauvaises réponses :

— Dans un premier temps, j'ai utilisé une fonction de perte non-supervisée (Cerisara, Caillon et Le Berre 2021) visant à séparer ces deux distributions de scores. L'idée derrière cette méthode est que si l'on connait la proportion d'exemples positifs et négatifs dans une base de données, on peut forcer le modèle à adopter une distribution des scores avec cette proportion. Dans le cas d'OpenBookQA et d'ARC, la proportion de bonnes et mauvaises réponses est facile à obtenir puisqu'il y a exactement 4 choix de réponse par question et donc une proportion de 75%/25% de réponses

correctes/distracteurs. Cette approche n'a pas eu de résultats probants et malgré l'efficacité de la fonction de perte ajoutée qui contraint bien la distribution à former 2 gaussiennes au lieu d'une seule, ces deux gaussiennes ne discriminent pas correctement les réponses. En effet, même ainsi les exemples positifs et négatifs (càd les bonnes et mauvaises réponses) obtiennent des scores qui sont répartis arbitrairement entre les deux distributions. Cette perte supplémentaire est donc correctement apprise mais ne régularise pas correctement.

— La deuxième approche fut l'utilisation d'une architecture siamoise. Une telle architecture vise à rapprocher les représentations de deux exemples similaires et d'éloigner des représentations correspondant à des exemples opposés. L'idée est la suivante : on dispose d'un modèle permettant pour une donnée d'entrée x de produire un vecteur de représentation de cette donnée $f(x)$ et on dispose également d'un ensemble d'entraînement contenant des couples d'exemples (x_i, x_j) avec un label indiquant si les deux exemples sont similaires ou non (ex : SNLI (Bowman et al. 2015)). On entraîne alors le réseau en minimisant $||f(x_i) - f(x_j)||$ si x_i et x_j sont similaires et en maximisant la même quantité s'ils ne le sont pas. Dans le cas du question-réponse, deux exemples similaires correspondent à soit deux réponses positives ou deux réponses négatives et deux exemples opposés à une réponse positive avec une réponse négative. L'idée est donc de rajouter entre chaque itération du processus d'entraînement, une ou plusieurs itérations en suivant le principe de l'architecture siamoise. J'ai expérimenté avec plusieurs versions de ce modèle et en particulier des applications de cette erreur à différents niveaux : au niveau des vecteurs de représentation de BERT et au niveau des scores scalaires eux même pour pousser le modèle à produire des scores. Dans cette dernière situation, l'objectif était de directement contraindre les scores à former deux clusters de bonnes et mauvaises réponses. Cette deuxième méthode est peu prometteuse dans le sens ou l'apprentissage est assez instable et ne parvient pas à former les clusters correctement. Augmenter le poids de cette partie de l'erreur aboutit systématiquement à une perte de performance sur la tâche de question-réponse.

2.7.2. Réduction adversarielle des biais

À la suite des résultats peu prometteurs des méthodes utilisant des régularisateurs, j'ai expérimenté avec une autre méthode utilisant le principe des réseaux de neurones adversariaux (Goodfellow, Pouget-Abadie et al. 2014). Déterminer un-à-un quels sont les biais présents dans une base de données est une tâche qui si toutefois s'avère possible reste plutôt pénible. On souhaiterait donc une méthode permettant de réduire ces biais sans avoir à les identifier. On sait cependant qu'un modèle entraîné sans qu'on lui donne la question n'apprend que des biais (puisqu'aucune information intéressante ne lui est donné). On peut donc utiliser un modèle entraîné de cette manière pour apprendre à un autre modèle à éviter ces biais.

Pour ce faire, on utilise deux ensembles d'entraînement distincts. Le premier contient des questions non-modifiées directement extraites de la base de données OpenBookQA. Et le deuxième contient uniquement des choix de réponses sans la question associée. A chaque itération de l'entraînement, un *mini-batch* de chaque ensemble sont envoyés au travers d'un unique modèle RoBERTa-Large qui produit les vecteurs de représentation correspondant. Ces représentations sont envoyées dans deux couches linéaires distinctes. La première reçoit les exemples complets (avec la question) et est entraînée ainsi que le modèle RoBERTa à correctement sélectionner la bonne réponse à la manière d'un apprentissage classique de RoBERTa pour le QCM. L'autre couche linéaire qui reçoit les exemples négatifs (sans la question) est également entraînée à sélectionner la bonne réponse en utilisant uniquement l'information contenue dans les réponses (et donc en utilisant exclusivement des biais). Au niveau de la connexion entre notre modèle RoBERTa-Large et ce deuxième sous-réseau, le gradient est inversé et RoBERTa est optimisé pour minimiser les résultats obtenus par ce sous-réseau apprenant des biais. Un schéma du réseau est présenté en figure 11. RoBERTa est donc entraîné à maximiser les résultats du sous-réseau apprenant avec la question tout en minimisant les résultats du sous-réseau apprenant sans la question. Le modèle est encouragé à incorporer le moins possible d'information sur le biais tout en passant quand même suffisamment d'information pour permettre au sous-réseau recevant la question de répondre correctement.

L'un des premiers problèmes que j'ai rencontrés avec cette architecture est que si on donne d'un côté des questions+réponses complètes et de l'autre uniquement des réponses, le modèle

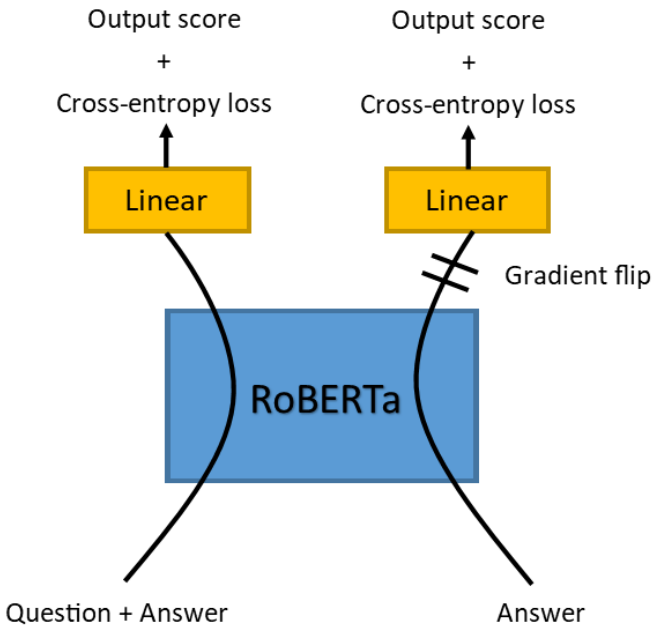


Figure 11 – Schéma de l'architecture d'entraînement du réseau de réduction adversarielle des biais.

réussit à distinguer les exemples positifs et négatifs en fonction de leur longueur (puisque les exemples contenant une question sont plus longs). Il contourne donc la régularisation en produisant de mauvais vecteurs de représentation lorsque la réponse est trop courte. C'est pourquoi il est nécessaire de remplacer la question dans les exemples négatifs par une question issue d'un autre exemple afin de donner aux exemples négatifs une forme plus convaincante. Cependant, avec cette modification, l'apprentissage des modèles devient très instable (un problème connu pour les architectures adversarielles). Cette instabilité des modèles génératifs couplée à la régularisation dégrade drastiquement les résultats obtenus par les modèles au point que ceux-ci deviennent à peine meilleurs qu'une sélection aléatoire. Cette piste qui ne me semblait pas prometteuse a donc été abandonnée au profit de méthodes de génération de questions que je présenterai plus dans cette thèse.

2.8. Conclusion

Dans ce chapitre, nous avons donc mis en évidence d'une part, qu'il existe des biais dans au moins certaines bases de données de question-réponse les plus couramment utilisées et

d'autre part que les modèles pré-entraînés (en particulier les modèles de type BERT) sont très sensibles à ces biais au point qu'une grande partie de leurs performances sur ces datasets puissent être expliquée par eux. Nous avons de plus identifié certains de ces biais comme la longueur des choix de réponse ou encore la présence de mots plus rares dans les réponses correctes. Nos résultats font écho à d'autres publications contemporaines à nos travaux qui mettent en évidence des biais et des comportement similaires sur d'autres tâches de NLP comme le NLI (*Natural Language Inference*) (Poliak et al. 2018).

La présence de ces biais est un problème puisqu'ils réduisent potentiellement les performances de généralisation des modèles en leur faisant apprendre une part de logique spécifique à un dataset en particulier et qui n'a rien à voir avec la logique de question-réponse que l'on voudrait intuitivement lui faire apprendre. Il est donc essentiel de prendre en compte ces biais à la fois lors de la création des bases de données et lors du design et de l'entraînement des modèles.

Dans un premier temps, j'ai expérimenté avec l'utilisation de mécanismes de régularisation afin de contraindre les modèles à n'apprendre que la partie utile de l'information contenue dans les bases de données de question-réponse. Cependant, ces méthodes rendent l'entraînement des modèles difficile et instable. Le problème majeur est qu'il est difficile pour les modèles de faire la différence entre une information utile et un biais. En effet, un biais n'est en réalité qu'une information à laquelle nous, humain, attachons une valeur particulière. Mais ce jugement qui vient du fait que nous comprenions d'un point de vue abstrait la tâche à laquelle nous avons affaire est inaccessible (du moins pour le moment) aux modèles d'intelligence artificielle. De plus, certaines de ces informations qui seraient considérées comme des biais pour une tâche donnée peuvent en réalité s'avérer utiles dans le contexte d'une autre tâche connexe.

Les chapitres suivants explorent donc différentes manières de contourner ce problème de biais dans les données. Tout d'abord, le chapitre 3 explore l'idée d'utiliser certains biais comme une source d'information applicable à une tâche connexe. Le chapitre 4 présente une méthode de génération automatique de questions pour attaquer à la source le problème des biais en créant dès le départ des bases de données moins biaisées. Enfin, le chapitre 5 abordera l'idée d'effectuer l'entraînement des modèles sur plusieurs bases de données variées en même temps afin de noyer les biais présents dans chacune de ces bases de données.

Chapitre 3

Extraction d'information à partir du biais

Dans ce chapitre, je montre qu'il est possible d'utiliser certains des biais propres aux annotateurs présents dans une base de données de résumé automatique. Ces biais qui résultent de la façon d'écrire de chaque annotateur peuvent être utilisés pour apprendre à diversifier les résumés générés par nos modèles.

Dans ce chapitre, je discute de la manière dont nous pouvons utiliser les biais qui sont présents dans certaines bases de données comme une source de connaissance et de variabilité. En effet, certains biais que l'on rencontre dans les bases de données résultent de l'annotation et sont donc imputables à l'annotateur ou aux annotateurs qui ont créé la base de données (comme mis en lumière dans le chapitre 2). Ceci implique que ces biais, loin d'être insignifiants, peuvent être révélateurs de la façon d'écrire des annotateurs. Si on prend l'exemple du résumé automatique (qui fera l'objet des expériences qui vont suivre), dans lequel on demande à un modèle de produire un texte plus court à partir d'une phrase ou d'un paragraphe source, il est tout à fait probable que plusieurs résumés soient possibles pour une même source. On peut imaginer par exemple que certains annotateurs produisent des résumés plus courts que d'autres ou bien que certains vont porter plus d'attention sur une partie ou une autre d'un texte. Lorsque ceci est le cas, plusieurs manières de résumer peuvent se retrouver dans une même base de données et les modèles risquent de converger vers une façon de résumer intermédiaire qui ne correspond au final à aucun des types proposés dans la base de données initiale. Si on se concentre uniquement sur la tâche de résumé et que l'on

demande à un modèle de produire un seul et unique résumé pour chaque phrase, ces différents types de résumés possibles vont agir, dans le contexte de cette tâche, comme des biais dans les données : si un seul type d'écriture est présent dans la base de données, il va biaiser l'apprentissage de nos modèles en tirant les résumés générés vers une distribution qui n'est pas représentative de la distribution réelle des résumés. De plus, si plusieurs types d'écritures sont présents, ils vont rendre les modèles plus confus et potentiellement ralentir l'apprentissage. Cependant, ce qui est un biais pour une tâche donnée peut être une information utile pour une autre tâche (connexe ou non). En effet, si dans le cas du résumé automatique, on considère la tâche comme une tâche de modélisation des différentes manières de résumer un texte, alors les différentes façons d'écrire que l'on retrouve dans les bases de données peuvent être utiles. Si elles sont prises en compte par les modèles, elles peuvent servir à apporter de la variabilité dans les sorties générées par ces derniers. On pourrait donc entraîner nos modèles à reconnaitre les différents styles d'écriture présents dans une base de données et à générer soit à la demande (contrôlabilité) soit aléatoirement (variabilité) des sorties variées. De plus, lorsqu'il existe une variabilité importante dans les bases de données, cela peut rendre la tâche plus difficile pour des modèles d'apprentissage automatique, et donc voir la tâche sous un autre angle peut permettre de réduire la confusion des modèles pour permettre un apprentissage moins bruité. Dans les sections qui suivent, je détaille une architecture d'encodeur-multidécodeur ainsi qu'une méthode d'entraînement permettant de prendre en compte cette variabilité naturelle qui se trouve dans une base de données de résumé automatique pour créer un modèle capable de générer plusieurs types de résumé. Cette étude a fait l'objet d'une publication à la conférence ESANN 2020 (Le Berre et Cerisara 2020). Il est important de noter que ces travaux ont été réalisés au tout début de mon doctorat (fin 2018 – début 2019). Les modèles et architectures état de l'art ont donc beaucoup évolués depuis avec notamment l'apparition de modèles pré-entraînés comme BERT, RoBERTa ou GPT qui ont rapidement remplacé les modèles basés sur des architectures LSTM (Hochreiter et Schmidhuber 1997b) qui étaient utilisés jusque-là. Cependant, la méthode qui est présentée dans ce chapitre est suffisament générale et indépendante de l'architecture utilisée pour pouvoir imaginer une application à ces nouveaux modèles état de l'art.

3.1. Introduction

Dans ce chapitre je parlerai donc du résumé automatique comme un exemple de tâche dans laquelle plusieurs styles d'écriture peuvent se trouver dans un même dataset. Bien que cette variabilité puisse potentiellement être néfaste pour l'apprentissage de la tâche de résumé, nous verrons qu'il est possible de capturer cette variabilité pour permettre à des modèles de générer plusieurs types de résumés.

Deux principaux types d'approches sont couramment utilisés pour résumer automatiquement le texte : d'une part, des approches extractives qui génèrent des résumés à l'aide d'un sous-ensemble de mots du texte original. D'autre part, des approches abstractives qui génèrent un résumé avec de nouveaux mots et une nouvelle structure syntaxique. Le résumé abstractif imite de manière plus convaincante la façon dont un annotateur humain écrirait réellement des résumés, en paraphrasant et en exploitant la richesse du langage naturel. Ainsi, dans un résumé, seuls quelques noms propres sont entièrement déterminés par la phrase originale, tandis que le reste des mots, ainsi que les structures syntaxiques choisies, dépendent fortement du style d'écriture de la personne qui rédige le résumé. Les systèmes séquence à séquence traditionnels ne parviennent pas à capturer cette variabilité et produisent généralement un résumé unique qui minimise l'erreur moyenne sur tous les styles d'écriture possibles dans le corpus d'entraînement.

Une façon courante d'apporter de la variabilité dans les sorties du système consiste à utiliser une représentation intermédiaire modélisée sous la forme d'une distribution de probabilité, comme avec les auto-encodeurs variationnels (VAE) (Kingma et Welling 2014). De cette manière, une fois le modèle entraîné, il suffit en théorie d'échantillonner des valeurs aléatoires pour ce vecteur de représentation intermédiaire et obtenir ainsi autant de résumés différents que l'on souhaite. Cependant, l'échantillonnage aléatoire ne permet pas de contrôler les propriétés des résumés obtenus et mes expérimentations montrent qu'en pratique, les VAE classiques ne génèrent que peu de variations réelles et que la plupart des différences ne portent que sur quelques mots. Certaines approches de contrôle des VAE (Guo, Y. Du et Zhao 2021) permettent tout de même d'augmenter cette variabilité ainsi que de la rendre contrôlable au prix d'avoir à manuellement prédéfinir un certain nombre de critères contrôlables.

Je propose une approche différente pour capturer et reproduire cette variabilité, en permettant à mon réseau de produire directement plusieurs résumés simultanément. Intuitivement, mon modèle est basé sur un réseau séquence à séquence standard avec attention. Cependant, là où un modèle séquence à séquence est constitué d'un encodeur produisant une représentation vectorielle intermédiaire et d'un décodeur utilisant cette représentation pour produire un résumé, mon modèle contient plusieurs décodeurs agissant en parallèle pour produire chacun leur propre résumé. L'ensemble du modèle est entraîné à l'aide d'un algorithme proche de l'algorithme Espérance-Maximisation (EM) afin que chaque décodeur se spécialise en capturant un style d'écriture différent à partir du corpus d'entraînement. Cette approche est similaire aux méthodes de type "*mixture of experts*" (Masoudnia et Ebrahimpour 2014 ; Yuksel, Wilson et Gader 2012) à la différence qu'au lieu de travailler ensemble pour produire une seule sortie, mes décodeurs génèrent chacun un résumé en essayant de couvrir au mieux l'espace des possibles.

3.2. Etat de l'art

La compression ou résumé de phrases est une tâche essentielle du traitement du langage naturel (TAL) proche de la génération de titres. Certains travaux se sont concentrés sur la structure syntaxique et les règles de réécriture comme dans (Dorr, Zajic et Schwartz 2003) où les techniques statistiques de traduction automatique (Banko, Mittal et Witbrock 2000).

Rush, Chopra et Weston (2015) ont introduit un nouvel ensemble de données pour la compression de phrases extraites du Gigaword anglais annoté (Napoles, Gormley et Van Durme 2012) Ils ont ensuite proposé un modèle composé d'un encodeur convolutif et d'un décodeur avec attention inspiré de Bengio, Ducharme et Vincent (2000). Des travaux ultérieurs (Chopra, Auli et Rush 2016) ont montré que le remplacement du modèle de langue de base par un réseau de neurones récurrent améliore les performances du modèle. Nallapati, Xiang et B. Zhou (2016) ont encore amélioré le modèle, finalisant sa transformation en un réseau séquence à séquence complet avec encodeur et décodeur LSTM (Long Short-Term Memory) (Hochreiter et Schmidhuber 1997b). En outre, ils ont inclus des informations supplémentaires (telles que les catégories grammaticales) dans

les entrées du modèle. Plusieurs améliorations supplémentaires de ces modèles ont été proposées : entre autres, Ayana et al. (2016) utilisent un “Minimal Risk Training” et Q. Zhou et al. (2017) ont proposé un mécanisme d'encodage sélectif pour contrôler le flux d'informations envoyé au décodeur. Plus récemment, Gehring et al. (2017) ont introduit un modèle entièrement convolutif avec attention qui obtient de bonnes performances pour le résumé de phrases. Banijamali, Ghodsi et Poupart (2017) ont également proposé un modèle génératif basé sur EM, étroitement lié à notre proposition. Cependant, leur algorithme utilise un clustering explicite, tandis que mon modèle regroupe implicitement les données d'entraînement un exemple après l'autre. En outre, je me concentre sur la synthèse de texte tandis que Banijamali, Ghodsi et Poupart (2017) se concentrent sur les tâches de génération d'images.

3.3. Modèles

3.3.1. Modèle de référence

Le modèle de référence que j'utilise à des fins de comparaison est un modèle séquence à séquence avec mécanisme d'attention. Chaque mot w_i de la séquence d'entrée $(w_i)_{1\leq i\leq T}$ est donné à une couche d'embedding $x_i = \text{emb}(w_i)$ qui produit une représentation vectorielle puis à un encodeur LSTM bidirectionnel (Hochreiter et Schmidhuber 1997b) qui produit une séquence de vecteurs d'état $(h_i^{\text{enc}})_{1\leq i\leq T}$.

Le dernier vecteur de cette séquence h_T^{enc} est donné au décodeur (lui aussi un réseau LSTM mais unidirectionnel cette fois), qui produit une nouvelle séquence de vecteurs d'état $(h_j^{\text{dec}})_{1\leq j\leq \tau}$ à partir de laquelle est produit un résumé $(\hat{w}_j)_{1\leq j\leq \tau}$.

Comme le décodeur est un réseau LSTM, il faut fournir au réseau une entrée additionnelle à chaque étape de la génération. Pendant l'entraînement, cette entrée j^{th} à l'étape j est le mot cible précédent (“teacher forcing”) $x_j^{\text{dec}} = \text{emb}(y_{j-1})$. Au moment du test, c'est le mot précédemment généré $x_j^{\text{dec}} = \text{emb}(\hat{w}_{j-1})$ qui est donné au décodeur à chaque étape. La longueur du résumé τ est déterminée au moment de l'exécution lorsque le token spécial $\hat{w}_j =$ EOS (End Of Sentence) est généré. A chaque étape de décodage j, un vecteur d'attention de taille fixe c_j est calculé comme suit (Bahdanau, Cho et Bengio 2015) :

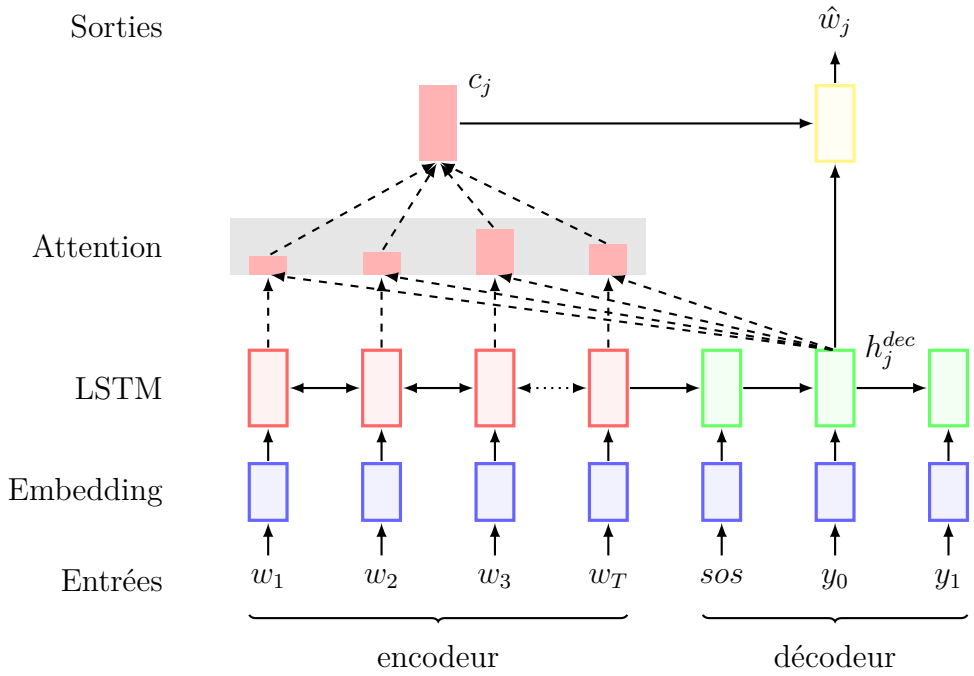


Figure 12 – Schéma du modèle encodeur-décodeur standard

$$a_j(i) = \frac{\exp(\text{score}(h_i^{\text{enc}}, h_j^{\text{dec}}))}{\sum_k \exp(\text{score}(h_k^{\text{enc}}, h_j^{\text{dec}})}$$

$$\text{score}(h_i^{\text{enc}}, h_j^{\text{dec}}) = (h_i^{\text{enc}})^T \cdot h_j^{\text{dec}}$$

$$c_j = \sum_{i=1}^{T} a_j(i) \times h_i^{\text{enc}}$$

Le vecteur c_j est alors concaténé à la sortie du décodeur h_j^{dec} et donné à une couche linéaire pour produire le mot généré $\hat{w}_j$.

3.3.2. Modèle seq-to-Nseq

Le modèle proposé améliore le modèle séquence à séquence classique en générant simultanément plusieurs résumés à l'aide de plusieurs décodeurs en parallèle :

$$\begin{aligned} \hat{w}^1_{1\ldots\tau_1} &= \text{dec}_1(h_T^{\text{enc}}) \\ \ldots &\quad \ldots \\ \hat{w}^n_{1\ldots\tau_n} &= \text{dec}_n(h_T^{\text{enc}}) \end{aligned}$$

Le modèle peut également être considéré comme plusieurs modèles séquence à séquence en parallèle avec des poids partagés pour l'encodeur, la couche d'embedding et l'attention. L'entraînement conjoint de l'encodeur et des décodeurs est réalisé avec une adaptation de l'algorithme Espérance-Maximisation, où, lors de l'apprentissage, un seul décodeur est échantillonné avec une variable aléatoire latente $z \sim \text{Catégorique}(n)$. Cela correspond à une version dure de l'algorithme EM, où la distribution a posteriori de z est approximée par un Dirac. Ainsi,

$$p(z|w_{1:T},y_{1:\tau}) \propto p(y_{1:\tau}|z,w_{1:T})p(z|w_{1:T})$$

et en supposant que tous les décodeurs sont équiprobables :

$$\hat{z} = \arg\max_z p(z|w_{1:T},y_{1:\tau}) \quad (1)$$

$$= \arg\min_z -\log p(y_{1:\tau}|z,w_{1:T}) \quad (2)$$

qui est le minimum de la perte de log-vraisemblance négative calculée à la sortie de chaque décodeur pendant l'apprentissage.

L'algorithme alterne donc entre :

— Espérance : faire un passage dans le réseau et sélectionner $\hat{z}$ comme dans l'équation 2.

— Maximisation : faire la backpropagation dans l'encodeur et le décodeur sélectionné $\text{dec}_{\hat{z}}$ et mettre à jour les paramètres.

Cet algorithme est assuré de converger vers un optimum local, qui peut être interprété comme un regroupement des résumés en plusieurs ensembles. Ce regroupement est réalisé selon un critère inconnu a priori, qui est automatiquement découvert lors de l'entraînement. Il peut s'agir par exemple d'un critère lexical, qui regroupe des résumés basés sur un sous-ensemble de mots préférés, ou un critère thématique, ou encore un critère combiné syntaxique-lexical qui représente divers styles d'écritures.

Après l'apprentissage, chaque décodeur s'est spécialisé pour générer un type de résumé spécifique. L'essentiel est que ce critère n'est pas défini a priori, mais plutôt automatiquement choisi pour être représentatif de la variété des styles qui apparaissent dans le corpus d'entraînement. Lorsqu'ils sont appliqués à des cas d'utilisation réels, les types de résumés appris peuvent être analysés (voir Section 3.5.2), et des résumés personnalisés peuvent être générés en choisissant un décodeur selon les préférences de l'utilisateur. j'implémente mon

modèle en utilisant un réseau séquence à séquence basique mais la méthode est applicable à des architectures encodeur-décodeur plus complexes.

Afin d'augmenter la différence entre les styles des sorties des décodeurs, j'ai également expérimenté une pénalité pour les résumés trop proches l'un de l'autre entre deux décodeurs. Cette pénalité est basée sur la perte "Cross Entropy" entre les sorties des décodeurs :

$$pénalité = \sum_{i,j,i\neq j} \max\left[\alpha - \frac{1}{\tau}\sum_{t=1}^{\tau} D_{KL}\left(\hat{w}_t^i || pred(\hat{w}_t^j)\right), 0\right]$$

où α est un hyperparamètre et $pred(.)$ représente la transformation suivante :

$$pred(\hat{w}_t^j)(k) = \begin{cases} 1 & \text{si } \hat{w}_t^j(k) = \underset{l}{max}\left(\hat{w}_t^j(l)\right) \\ 0 & \text{sinon} \end{cases}$$

Cette pénalité incite le système à générer des résumés aussi différents que possible entre deux décodeurs. Ce système est appelé "PEN". Une autre approche testée pour augmenter cette différence, nommée "NRI", consiste à entraîner le modèle avec des paramètres de décodeurs qui sont initialisés à partir de points distants dans l'espace des paramètres. Une telle initialisation est également utile lorsque l'on veut inciter les décodeurs à apprendre un style d'écriture spécifique. Par conséquent, dans le cas où on utilise 2 décodeurs, notre modèle NRI est initialisé en divisant les phrases d'apprentissage en deux groupes : les phrases les plus courtes d'un côté et les plus longues de l'autre par exemple. Chacun des deux décodeurs est alors pré-entraîné sur un groupe. PEN et NRI ont en pratique montré une augmentation de la divergence entre les sorties des décodeurs mais cela n'a entraîné aucune amélioration des performances selon les métriques habituelles utilisées en résumé automatique.

3.4. Protocole expérimental

3.4.1. Dataset

Ces modèles sont évalués sur la base de données Gigaword anglais 5e édition, qui contient des articles d'actualité provenant de différentes sources. J'utilise la version de résumé de

Phrase d'entrée :

thailand 's national carrier thai airways international said wednesday it has signed a code-sharing agreement with bahrain-based gulf air to meet rising demand for air travel.

Résumé cible :

thai airways gulf air sign code-sharing pact

Figure 13 – Un exemple de résumé dans la base de données Gigaword.

phrase décrite dans (Rush, Chopra et Weston 2015) [1]. Cet ensemble de données est construit en associant la première phrase des articles de Gigaword à leurs titres. Les paires première phrase/titre sont alors utilisées comme paires phrase/résumé. L'ensemble d'entraînement est composé de 3,8 millions de paires. Nous utilisons le découpage entraînement/développement/test fourni par (Rush, Chopra et Weston 2015). Les ensembles de développement et de test sont constitués de 2 000 paires chacun. La longueur moyenne des phrases sources est de 31 mots et la longueur moyenne des résumés cibles est de 8 mots. Un exemple de phrase et du résumé associé est présenté en figure 13.

3.4.2. Evaluation

La métrique d'évaluation est la norme ROUGE (C.-Y. Lin 2004) largement utilisée dans le résumé de texte. Je rapporte les résultats pour ROUGE-1 (chevauchement des unigrammes), ROUGE-2 (chevauchement des bigrammes) et ROUGE-L (chevauchement le plus long des sous-chaînes communes). De plus, puisque l'objectif est d'augmenter la variabilité des résumés générés, je rapporte également le pourcentage de résumés différents (entre les deux décodeurs par exemple) et la distance d'édition moyenne entre les résumés. J'effectue de plus une analyse qualitative.

3.4.3. Détails d'implémentation

Tous les hyperparamètres, à l'exception du nombre d'époques et du taux d'apprentissage (qui sont réglés sur l'ensemble de développement), sont copiés du package OpenNMT (G. Klein et al. 2017) : notre encodeur est un réseau LSTM bidirectionnel avec 2 couches et 250 dimensions (500 après concaténation entre les deux directions). Tous les décodeurs sont des LSTM unidirectionnels à 2 couches avec une taille de représentation de 500. Les 50 000

1. Extrait de la version annotée de Gigaword proposée dans (Napoles, Gormley et Van Durme 2012).

	R1 (F)	R2 (F)	RL (F)
ABS Rush, Chopra et Weston 2015	29.55	11.32	26.42
ABS+ Rush, Chopra et Weston 2015	29.76	11.88	26.96
RNN MLE Ayana et al. 2016	32.67	15.23	30.56
RNN MRT Ayana et al. 2016	36.54	16.59	33.44
ConvS2S Gehring et al. 2017	35.88	17.48	33.29
Var. Enc-Dec (VED)	35.29	16.77	32.83
seq-to-seq (baseline)	35.04	16.47	32.59
seq-to-2seq - output 1	34.90	15.81	32.27
seq-to-2seq - output 2	34.80	15.59	32.18
seq-to-3seq - output 1	32.51	13.65	29.37
seq-to-3seq - output 2	34.47	15.35	31.80
seq-to-3seq - output 3	34.93	15.77	32.56

Tableau 7 – ROUGE F-scores pour les modèles proposées et l'état de l'art sur Gigaword.

mots les plus fréquents du vocabulaire sont encodés dans des vecteurs d'embedding à 500 dimensions. Le modèle est entraîné pendant 15 époques en utilisant une descente de gradient stochastique avec un taux d'apprentissage initial de 1 divisé par 2 à chaque époque après l'époque 8. La taille du mini-batch est de 64. Pour le modèle PEN, je prends $\alpha = 0{,}5$ sur la base des meilleurs résultats sur l'ensemble de développement. Je teste deux versions de mon modèle avec $n = 2$ et $n = 3$ décodeurs respectivement et je teste les pénalités PEN et NRI avec $n = 2$ décodeurs.

3.5. Résultats

3.5.1. Résultats quantitatifs

Le tableau 7 compare les scores ROUGE de mes modèles avec des travaux connexes sur l'ensemble de test de Gigaword. Chaque décodeur individuel du modèle seq-to-Nseq obtient des résultats légèrement inférieurs à ceux obtenus par un modèle seq2seq standard, ce qui est attendu puisque chaque décodeur dans le modèle seq-to-Nseq est entraîné pour capturer seulement une fraction du corpus d'entraînement (la partie sur laquelle il est spécialisé). Je

	R1 (F)	R2 (F)	RL (F)
Baseline (2x training)	38.58	18.97	35.84
VED (2x sampling)	35.99	17.16	33.44
seq-to-2seq	39.48	19.49	36.68
seq-to-2seq+PEN	38.96	19.09	35.98
Baseline (3x training)	40.26	20.15	37.28
seq-to-3seq	41.26	20.77	38.09

Tableau 8 – ROUGE-1 (F), ROUGE-2 (F) and ROUGE-L (F) sur Gigaword quand on utilise un oracle pour choisir le meilleur candidat parmi plusieurs résumés générés.

reporte donc dans le tableau 8 aussi le ROUGE dans le cas où un oracle choisit, pour chaque phrase, la meilleure solution parmi $N \in \{2,3\}$ propositions. Ces propositions correspondent soit aux résumés proposés par les différents décodeurs (dans le cas du modèle proposé), soit à deux échantillonnages aléatoires (dans le cas du VAE), soit à des résumés obtenus en entrainant un même modèle plusieurs fois avec une initialisation aléatoire des paramètres initiaux. Bien sûr, de tels scores ROUGE ne sont pas comparables à l'état de l'art sur ce corpus, mais ils montrent que la variabilité des styles d'écriture est mieux couverte avec le modèle seq-to-Nseq qu'avec d'autre modèles.

Le tableau 9 indique le pourcentage de paires de résumés qui diffèrent d'au moins un mot (*diff*) et la distance d'édition moyenne entre deux propositions de résumés. Comme dans le tableau 8, chaque paire est obtenue soit par réapprentissage, rééchantillonnage (Variational Encoder-Decoder (VED)) ou bien correspond aux sorties de différents décodeurs. Bien que le VED puisse générer de nombreux résumés par rééchantillonnage, mes résultats montrent que la variabilité qui en résulte est beaucoup plus faible que lors d'un réapprentissage complet d'un modèle séquence à séquence non variationnel. A l'inverse, les décodeurs de mon modèle génèrent en moyenne des résumés bien plus variés.

3.5.2. Analyse qualitative

Les trois exemples de la figure 14 illustrent des différences typiques que l'on retrouve entre les résumés obtenus par les différents décodeurs du modèle seq-to-Nseq. Dans le premier exemple, les deux décodeurs génèrent un résumé sous la forme active et un résumé sous la forme passive. Dans le second exemple, on peut voir une différence notable dans la longueur

	diff (%)	edit distance
Baseline (2x training)	79.86	15.97
VED (2x sampling)	16.15	2.95
seq-to-2seq	89.44	21.77
seq-to-2seq+PEN	98.56	27.10
seq-to-2seq+NRI	96.26	26.48

Tableau 9 – Variabilité des résumés obtenus par différents systèmes. Plus les valeurs sont grandes, plus les résumés comparés sont différents l'un de l'autre.

des résumés tandis que dans le troisième exemple, les deux décodeurs mettent l'accent sur différentes informations.

Un point plus intéressant est que la manière dont chaque décodeur écrit ses résumés est consistante et chaque décodeur a un style qui lui est propre. Les trois exemples de la figure 15 illustrent les différences entre la voix passive et la voix active. Dans la majorité des cas, lorsqu'une telle différence est visible entre les deux décodeurs, le choix de chaque décodeur reste toujours le même. C'est-à-dire que le même décodeur génère la voix active tandis que l'autre se concentre sur la forme passive. Le processus est plutôt stable dans le sens où, après l'époque 13 pendant l'apprentissage, pour 80% des phrases, si le résumé produit par un décodeur N est le plus proche de la cible à l'époque 13 (et donc que pour cet exemple, c'est ce décodeur qui sera mis à jour), alors c'est ce même décodeur qui sera choisi pour toutes les époques successives.

3.6. Conclusion

Dans ce chapitre, je propose donc une manière de capturer une partie de la variabilité présente dans une base de données de résumé automatique. Le modèle proposé est une extension du modèle séquence à séquence LSTM avec attention standard capable de générer plusieurs résumés différents pour une même phrase d'entrée. Contrairement à d'autres méthodes comme (Fan, Grangier et Auli 2018) qui nécessitent d'identifier au préalable l'ensemble des propriétés que l'on souhaite pouvoir contrôler, ma méthode identifie automatiquement les différents styles d'écriture présents dans une base de données sans que ceux-ci

Phrase d'entrée :

Police arrested five anti-nuclear protesters Thursday after they sought to disrupt loading of French antarctic research and supply vessel, a spokesman for the protesters said

Résumé cible : Protesters target French research ship

Baseline : Five arrested in anti-nuclear protest

seq-to-2seq résumé 1 : Five arrested in anti-nuclear protest

seq-to-2seq résumé 2 : French police arrest five anti-nuclear protesters

Phrase d'entrée :

A young Syrian woman who was arrested last year on terrorism charges at the airport here had a map of us military facilities in turkey, a Canadian security official said Thursday.

Résumé cible : Canada investigates Syrian women with alleged ties to PKK

Baseline : Syrian woman arrested in Turkey had US military facilities

seq-to-2seq résumé 1 : Syrian woman arrested in Turkey on terrorism charges

seq-to-2seq résumé 2 : Canadian security official says Syrian woman had US military facilities in Turkey

Phrase d'entrée :

The head of the Russian-installed government in the breakaway republic of Chechnya narrowly survived a bomb attack Monday, the third assassination attempt against top Russian officials in two months.

Résumé cible : Head of UNK Chechen government survives bomb attack

Baseline : Chechen government survives bomb attack

seq-to-2seq résumé 1 : Chechen leader survives bomb attack

seq-to-2seq résumé 2 : Third assassination attempt in Chechnya

Figure 14 – Quelques exemples de résumés générés par le modèle seq-to-2seq.

Résumé 1 : Chinese pro-democracy activist arrested in Shanghai **Résumé 2 :** Chinese police arrest pro-democracy activist
Résumé 1 : Sudanese convicted of drug trafficking beheaded **Résumé 2 :** Saudi Arabia beheads Sudanese convicted of drug trafficking
Résumé 1 : Chinese fishing boat hijacked off east Africa **Résumé 2 :** pirates hijack Chinese fishing boat in Somalia

Figure 15 – Illustration de la cohérence avec laquelle chaque décodeur génère ses résumés.

ne doivent être définis ou contrôlés a priori. Plusieurs décodeurs/rédacteurs sont ainsi entraînés, chacun spécialisé dans son style de prédilection. L'analyse qualitative des résultats révèle que ma méthode capture une plus grande partie de la diversité présente dans la base de données et produit des résultats plus variés comparés aux VAEs par exemple.

Cette variabilité dans les styles des rédacteurs peut traditionnellement être considérée comme un biais car elle n'apporte pas en soit d'information factuelle sur la manière de résumer un texte et peut s'avérer gênante pour l'apprentissage des modèles. Cependant, je montre que dans le cadre d'une variation de la tâche initiale, ces biais peuvent être utiles. Dans notre cas, plutôt que de chercher à simplement résumer un texte, on cherche plutôt à rendre cette génération contrôlable par l'utilisateur afin que celui-ci puisse choisir le style de résumé qu'il trouve le plus agréable à lire.

Ces expériences montrent que dans certains cas, les biais présents dans les bases de données peuvent être utilisés par les modèles pour en extraire de l'information. Ce type de méthode bien qu'intéressante n'est bien évidemment pas applicable à tous les biais et dans la suite de cette thèse nous explorons diverses méthodes pour réduire la présence de biais ou atténuer leurs effets.

Bien sûr, de nombreuses autres avancées en matière de résumé automatique ont été présentées depuis la réalisation de ces travaux au début de mon doctorat et en tête de liste, l'arrivée de modèles pré-entraînés tels que BERT. Cependant, la méthode présentée dans ce chapitre est a priori indépendante de l'architecture utilisée et est applicable à d'autres modèles et il est donc naturel de tester le remplacement des LSTM que nous utilisons par des architectures plus récentes comme des *transformers* (pré-entraînées ou non).

Chapitre 4

Génération automatique de bases de données débiaisées

Dans ce chapitre, je montre qu'il est possible d'améliorer les performances de certains modèles de question-réponse multitâches pré-entraînés (UnifiedQA) sur de nouveaux domaines en les affinant avec des questions générées automatiquement. Cette méthode de génération automatique de questions permet d'éliminer à la source certains biais résultant d'une annotation humaine et de limiter le coup d'adaptation des modèles à de nouvelles bases de données.

4.1. Introduction

À la suite des expériences du chapitre 2, il est apparu qu'il serait probablement difficile d'empêcher l'apprentissage du biais par les modèles d'apprentissage profond. Une meilleure solution serait d'avoir accès dès le début à une base de données débarrassée de certains biais. Dans cette optique, j'ai choisi d'orienter ma recherche vers la génération non-supervisée de questions et la création de bases de données de question-réponse à choix multiples. Cette approche me permet de tenter de résoudre la tâche de question-réponse de manière non-supervisée et étudier quelles performances il est possible d'obtenir comparées à l'état de l'art supervisé. En plus de cette facette non-supervisée, cette approche permet également de résoudre les deux problèmes du question-réponse énoncés plus haut. Premièrement, il devient possible d'incorporer moins de biais dans les bases de données en adaptant le processus de génération. Par exemple, en utilisant un processus de génération qui utilise les

vraies réponses à d'autres questions pour générer les distracteurs, il est possible de s'assurer qu'il n'y a pas de différences morphologiques entre les réponses correctes et les distracteurs. Ce genre d'approche, où au moins une partie des distracteurs sont également les bonnes réponses à d'autres questions, est par exemple utilisé pour la création de la base de données CommonSenseQA (Talmor et al. 2019). De plus, une fois muni d'une méthode de création de questions efficace, il est également possible de créer à la volée, des questions pour des domaines spécifiques en réglant ainsi le problème du manque de données.

Ce chapitre se découpe en deux sous-parties principales. Premièrement, j'ai travaillé à la génération d'une base de données de questions dans le but d'entraîner un modèle non-supervisé sur la base de données OpenBookQA. L'avantage qu'offre OpenBookQA pour mon problème de génération de questions est que cette base de données comprend en plus des questions un ensemble de faits de connaissances générales sous la forme de phrases simples à partir desquelles on peut créer des questions. Dans cette première partie, je présente donc une méthode permettant de générer des questions à partir de cet ensemble de faits. J'utilise ensuite cet ensemble de questions pour entraîner un modèle non-supervisé sur OpenBookQA. Cette approche a déjà montré des résultats intéressants en considérant son caractère non-supervisé mais encore loin de l'état de l'art supervisé sur cette tâche.

Pendant la réalisation de ces premières expériences sur des approches non-supervisées, des modèles de question-réponse généraux et multitâches ont vu le jour (Khashabi, Min et al. 2020). Ces modèles qui sont entraînés sur de nombreuses bases de données de différents types (choix multiples, extractif, abstractif, etc.) et de différents domaines sont souvent déjà très performants lorsqu'appliqués à de nouvelles bases de données qu'ils n'ont pas vues pendant leur entraînement. Dans ce contexte, il m'est apparu que des approches non-supervisées ne faisaient plus de sens si de meilleurs résultats peuvent être obtenus en entraînant un modèle sur un ensemble de bases de données déjà existantes. En revanche, mes expériences préliminaires ont montré que lorsqu'on applique ces modèles généraux à des nouvelles bases de données, leurs performances peuvent être encore améliorées en affinant ces modèles avec des données annotées issues de celles-ci. Cette différence entre les performances de modèles généraux affinés et non-affinés sur la tâche considérée est alors une cible de choix pour mes méthodes de génération de questions. Dans la deuxième partie de ce chapitre, je m'attaque donc à la génération d'une base de données de questions afin d'affiner un modèle multitâche

– UnifiedQA (Khashabi, Min et al. 2020) – sur de nouvelles bases de données qu'il n'a pas vues pendant son entraînement. Cette partie a fait l'objet d'une publication à ACL 2022 (Le Berre, Cerisara et al. 2022).

Finalement, dans une troisième partie, je présente une méthode non-supervisée de génération de questions qui n'a pas été retenue pour la suite de mes expériences mais qui néanmoins produit des résultats intéressants. Je présente donc cette méthode ici car elle pourrait faire l'objet de recherches plus approfondies par la suite.

4.2. Etat de l'art

4.2.1. La génération de questions

Du côté de la génération supervisée de questions, plusieurs solutions existent. X. Du, Shao et Cardie (2017) par exemple utilisent la base de données SQUAD. Dans SQUAD, puisque la tâche consiste en l'extraction de la réponse dans un texte, il est possible pour chaque question d'extraire la phrase qui contient la réponse et ainsi de créer des couples phrase/question permettant d'apprendre de manière supervisée à un modèle à produire la question à partir de cette phrase. C'est l'idée développée dans (X. Du, Shao et Cardie 2017) où le modèle utilisé est un modèle séquence à séquence construit à partir d'OpenNMT (G. Klein et al. 2017). La tâche devient plus complexe lorsqu'on cherche à générer des questions de manière non-supervisée. Bien sûr, une des possibilités pour générer des questions à partir d'une base de données de phrases affirmatives est d'utiliser des règles de réécriture. C'est le cas dans (Das et al. 2016) par exemple. Certaines solutions d'apprentissage profond existent également et notamment des solutions inspirées de méthodes de traduction automatique non-supervisée. Dans (Lewis, Denoyer et Riedel 2019), le modèle utilisé pour obtenir ce résultat est un ensemble de deux modèles séquence à séquence $f(\cdot)$ et $g(\cdot)$, l'un apprenant à former des questions à partir de phrases à trous et l'autre apprenant la tâche opposée. Les deux modèles sont entraînés conjointement afin d'un côté de minimiser l'écart entre une phrase x et la double transformation $g(f(x))$ et de l'autre de minimiser parallèlement l'écart entre une question y et $f(g(y))$. Cette méthode de génération de questions est utilisée sur la base de données SQuAD (Rajpurkar, Zhang et al. 2016). Fabbri et al. (2020) et Z. Li et al. (2020) améliorent encore l'état de l'art non-supervisé sur SQuAD et montrent que des

méthodes basées sur des règles de réécriture simples peuvent dans certains cas surpasser les méthodes neuronales.

4.2.2. La sélection de distracteurs

Quelques approches existent aussi pour la génération supervisée de distracteurs. C. Liang et al. (2018) par exemple, proposent d'utiliser différents algorithmes de classement afin de choisir les distracteurs à associer à une question donnée. Les auteurs utilisent notamment des approches de régression logistique et de forêts aléatoires en les appliquant sur un certain nombre de *features* comme les similarités dans les POS tags ou la distance d'édition entre la réponse correcte et les distracteurs. (Ren et K. Q. Zhu 2021) est un autre article traitant de la génération de distracteurs, cette fois en utilisant des bases de connaissances (ex : WordNet ou Probase). Leur idée est que les distracteurs pour une question sont souvent des entités simples pouvant se retrouver dans ce genre de base de connaissances et que ces distracteurs sont souvent proches de la réponse correcte dans le graphe (par exemple reliés par un même nœud parent). Ils extraient ainsi un ensemble possible de distracteurs qui sont ensuite classés en utilisant encore une fois un ensemble de *features* extraits des questions et de chaque distracteur. Pour la génération de distracteurs, il n'existe à ma connaissance pas d'approche non-supervisée.

4.3. Génération de questions pour des approches non-supervisées

Afin d'évaluer la faisabilité d'une approche impliquant la génération non-supervisée de questions, j'ai réalisé des expériences à l'aide de modèles simples. J'ai utilisé les deux bases de données de faits fournies avec OpenBookQA. Mises ensemble, on obtient un ensemble de plus de 6000 faits à partir desquels on génère des questions. L'intérêt de prendre ces faits est que ceux-ci sont certains de contenir le même type d'information que les questions dans OpenBookQA. Je simule ainsi une extraction parfaite de textes du même domaine que les questions. Cette base de données de phrases parfaitement reliées au domaine des questions me permet d'évaluer une borne supérieure des performances qu'il serait possible d'obtenir avec une selection moins parfaite des phrases (une sélection à partir de Wikipédia par exemple).

4.3.1. Processus de génération

OpenBookQA et plus généralement les bases de données de question-réponse sont constituées de deux types de questions. D'une part, des questions "bien formées" commençant le plus souvent par un mot en "*Wh*" (*What*, *Where*, *When*, etc.) et se terminant par un point d'interrogation et d'autre part, des questions de complétion de phrases pour lesquelles la question elle-même est une phrase tronquée d'une partie (pour OpenBookQA, le plus souvent la fin) et les réponses correspondent à cette partie manquante. Ce deuxième type de question est beaucoup plus facile à obtenir à partir de la base de données de faits et c'est donc par là que j'ai commencé. Ma première approche a donc été de couper ces phrases à un emplacement aléatoire pour former une question et une réponse associée. Puisque OpenBookQA est une tâche de QCM, il est nécessaire pour chaque question de proposer des distracteurs. Une sélection de distracteurs complètement aléatoire n'est pas très convaincante et produit des données d'entraînement trop faciles pour les modèles qui ont tendance à apprendre à différencier les bonnes réponses en utilisant seulement la grammaire des phrases puisque les réponses incorrectes sont souvent mal formées de ce point de vue. J'ai donc essayé d'affiner cette approche en sélectionnant des distracteurs dont le début de la séquence de *POS tags* (*Part Of Speech tags*) est la même que celle de la réponse correcte. Cette méthode est rapidement limitée puisque si on augmente le nombre de POS tags devant être identiques, il n'y a rapidement plus aucune séquence correspondante dans la base de données. De plus, cette méthode, bien qu'elle soit meilleure qu'une sélection aléatoire, ne garantit pas la cohérence grammaticale entre la question et les distracteurs.

J'ai par ailleurs travaillé sur l'arbre syntaxique des phrases. J'ai généré des questions de la manière suivante : lors du parcours de l'arbre, les phrases nominales, phrases verbales et phrases prépositionnelles sont marquées comme candidats potentiels pour devenir des réponses. A partir de cette liste de candidats, on élimine les phrases verbales qui ont pour verbe un auxiliaire ou un verbe d'état en utilisant une liste de noire des verbes d'état les plus courants. Les questions sont alors créées de deux manières différentes :

— Dans un premier cas, on remplace le candidat dans la phrase par une liste de remplacements possibles en fonction de son type. Par exemple, les phrases nominales sont remplacées par "*what*" ou "*which*". Dans le cas des phrases verbales, on remplace par la conjugaison appropriée de "*do what*" en fonction du temps du verbe dans la phrase

Plants are what ?	**A : producers**
	B : they
	C : iron
	D : a microscopic level
Temps rising is an increase...	**A : in temperature**
	B : in this alloy
	C : in trees
	D : on a microscopic level

Figure 16 – Deux exemples de questions générées automatiquement avec des distracteurs aléatoires.

verbale initiale. Un verbe au participe présent devient donc "*doing what*". Dans le cas des phrases prépositionnelles, le mot interrogatif dépend de la préposition elle-même (ex : "*where*" pour la préposition "*in*"). Cette méthode de remplacement ne garantit pas que la question créée sera toujours bien formée (par exemple des questions avec un mot intérrogatif à la fin de la phrase) mais ce genre de questions se retrouvent également dans OpenBookQA.

— La deuxième possibilité lorsque le candidat se trouve à la fin de la phrase est de le retirer complétement pour former une question de type complétion de phrase.

Pour chaque phrase dans la base de données de faits, on construit deux questions (une avec chaque méthode énoncée ci-dessus). On obtient donc environ 12 000 questions qui sont séparées en 90% de données d'entraînement et 10% de données de validation. Pour ces 12 000 questions, on sélectionne 3 distracteurs aléatoirement par question en choisissant des candidats ayant le même type syntaxique que la réponse (ainsi que la même forme du verbe pour les phrases verbales) afin de maximiser la cohérence grammaticale entre une question et les distracteurs associés. Ce processus de création de questions est entièrement non-supervisé. La figure 16 montre des exemples de questions et leurs distracteurs obtenus avec ce procédé.

Avec cette méthode de génération, les questions générées sont encore assez faciles à résoudre. Les distracteurs étant choisis au hasard, ils n'ont souvent aucun sens par rapport à la question, ce qui n'est souvent pas le cas dans une "vraie" base de données annotée. Je propose donc une méthode pour sélectionner des distracteurs plus difficiles pour les questions. Pour ce faire, j'utilise un modèle RoBERTa qui est dans un premier temps entraîné

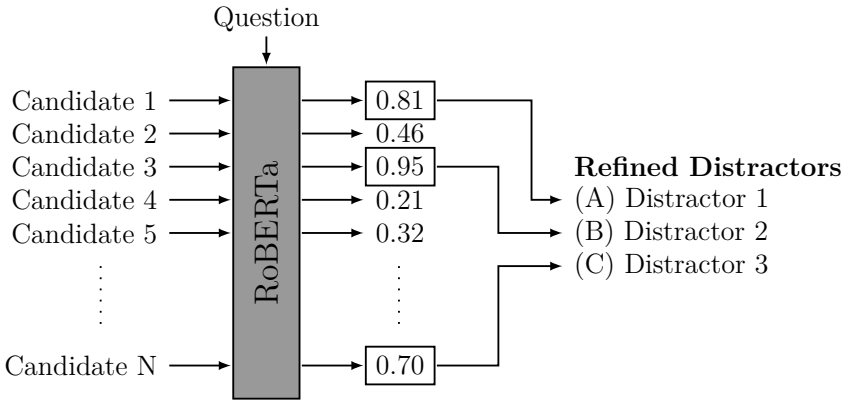


Figure 17 – Illustration de la méthode de raffinage des distracteurs. Pour chaque question, un ensemble de candidats est choisi aléatoirement puis RoBERTa attribue à chacun d'eux un score en fonction de leur proximité à la question. Finalement, les 3 candidats les plus proches de la question sont retenus pour devenir les 3 nouveaux distracteurs pour cette question.

en utilisant mes questions synthétiques avec des distracteurs aléatoires. Lors de cet apprentissage, RoBERTa apprend à différencier les bonnes réponses des distracteurs et donc d'une certaine manière, le modèle apprend ce qui fait qu'un choix de réponse est plausible comme réponse à une question donnée. Après avoir obtenu ce modèle entraîné, je l'utilise donc pour attribuer un score à un ensemble de candidats distracteur pour une question donnée (encore une fois, ces candidats distracteurs sont choisis parmi les réponses à des questions similaires). Evidemment, scorer tous les candidats distracteurs possibles pour toutes les questions serait bien trop coûteux en temps. De plus, plus le nombre de candidats est grand, plus il est probable d'obtenir des candidats distracteurs qui sont en réalité des bonnes réponses et je risque donc de sélectionner des distracteurs qui sont des réponses sémantiquement correctes. Pour chaque question, je sélectionne donc d'abord aléatoirement N candidats (N étant un hyperparamètre) parmi tous les distracteurs possibles puis j'utilise RoBERTa entraîné pour scorer ces N candidats distracteurs par rapport à la question selon que le modèle estime que ce candidat est un choix de réponse crédible ou non. Finalement, je choisis les 3 distracteurs ayant obtenus les meilleurs scores comme nouveaux distracteurs pour cette question. Ce processus est décrit en figure 17. Pour cette expérience, l'hyperparamètre N est fixé à 20 candidats distracteurs par question. Il faut noter qu'avec $N = 20$, le processus de re-sélection des distracteurs demande 20 passages dans RoBERTa par question dans la base de données synthétique. Les questions ainsi que leurs 3 nouveaux distracteurs forment un nouvel ensemble de 12 000 questions (figure 18) sur lequel on va affiner un nouveau modèle

Plants are what ?	**A : producers**
	B : an example
	C : a source
	D : aquatic life
Temps rising is an increase...	**A : in temperature**
	B : in altitude
	C : in warm weather
	D : over time

Figure 18 – Les deux exemples de questions montrés précédement avec leurs nouveaux distracteurs choisis parmi les plus difficiles à différencier par un modèle entraîné avec des discriminateurs aléatoires.

RoBERTa. Je compare également deux situations : l'une où les 3 distracteurs sont remplacés par les nouveaux et l'autre où on remplace uniquement 2 d'entre eux et où on garde le dernier sélectionné au hasard.

4.3.2. Résultats

Cet ensemble de données est testé en ajustant un modèle RoBERTa-Large et en testant le modèle ainsi obtenu sur OpenBookQA et ARC. Le modèle est entraîné en utilisant une optimisation "Adam" avec une vitesse d'apprentissage de 1×10^{-5}. Les résultats de ces expériences sont reportés en table 10. On observe que si j'entraîne ce modèle sur uniquement cet ensemble de questions générées automatiquement, j'obtiens des résultats modestes mais déjà intéressants si on considère que le processus est non-supervisé.

Les trois lignes du milieu de la table 10 montrent les résultats obtenus par différentes versions du système de sélection des distracteurs. On note que le raffinage des distracteurs permet une amélioration notable des performances par rapport à une base de données avec des distracteurs aléatoires. Ceci permet même de dépasser les 50% d'*accuracy* sur ARC et OpenBookQA. De plus, si on s'écarte du non-supervisé et que j'affine par la suite le modèle sur les données d'entraînement respectives d'OpenBookQA et ARC, j'obtiens une augmentation significative des performances obtenues par RoBERTa sur ces données d'entraînement seules ce qui montre que la génération de questions peut également être utile en tant que pré-entraînement avant de passer à des données annotées.

	OBQA	ARC
apprentissage supervisé	65.2	52.6
distracteurs aléatoires*	47.8	47.5
distracteurs v1*	**51.2**	48.0
distracteurs v2*	49.8	**50.0**
distracteurs v2 + affinage	70.6	62.7

Tableau 10 – Résultats en termes d'*accuracy* des approches non-supervisées. Ce tableau montre une comparaison des scores d'*accuracy* obtenus entre une méthode où les distracteurs sont choisis au hasard (distracteur aléatoire), une méthode où tous les distracteurs sont resélectionnés parmi ceux qui sont les plus difficiles à écarter pour un premier entraînement (distracteur v1) et une méthode où on ne reselectionne que 2 distracteurs en laissant le dernier sélectionné au hasard (distracteur v2). La première ligne montre pour comparaison les résultats obtenus par un modèle RoBERTa-Large affiné de manière supervisée sur la tâche. La dernière ligne montre les résultats obtenus si on entraîne RoBERTa d'abord sur la base de données non-supervisée puis qu'on affine sur les données spécifiques à la tâche. Le symbole * indique les approches qui sont non-supervisées (qui n'utilisent pas les données spécifiques aux tâches).

4.4. Affinage non-supervisé de modèles généraux

Cette première méthode de génération de questions et des réponses associées a montré de bons résultats pour permettre l'entraînement non supervisé de modèles de question-réponse. Cependant, même si cette approche, utilisée comme pré-entraînement, permet une augmentation importante des performances obtenues par un modèle lorsqu'on utilise uniquement les données spécifiques à une base de données (ex : si on utilise uniquement les données annotées d'OpenBookQA), elle est rapidement supplantée par des modèles entraînés sur plusieurs bases de données. En effet, pour le question-réponse à choix multiples, il est courant dans l'état de l'art de voir des modèles qui sont par exemple entraînés sur à la fois ARC, OpenBookQA et RACE. Cet entraînement plus général permet aux modèles d'accroître l'information engrangée en profitant des différences de domaines et de sujets abordés dans chaque dataset. Les modèles obtenus ont souvent également une plus grande capacité de généralisation à de nouvelles tâches ou domaines dû au fait que la diversité de données d'apprentissage permet de diminuer le surapprentissage. C'est cette même approche d'apprentissage sur plusieurs bases de données à la fois qui a donné naissance à UnifiedQA (Khashabi, Min et al. 2020). UnifiedQA est un modèle construit à partir du modèle T5 et entraîné sur un

grand nombre de bases de données de question-réponse de différents types (choix multiples, vrai/faux, questions ouvertes, ...). UnifiedQA a de plus une bonne capacité de généralisation à des bases de données et à des domaines qu'il n'a pas vus lors de son entraînement résultant en des performances sur des nouvelles bases de données supérieures à ce qui est obtenu par des méthodes non supervisées. Cette observation rend les méthodes non-supervisées beaucoup moins intéressantes. Cependant, on remarque que les scores obtenus par UnifiedQA sur une nouvelle base de données peuvent être améliorés lorsque l'on affine ce premier avec des données annotées spécifiques à la tâche. Ceci semble indiquer que malgré la grande capacité de généralisation d'UnifiedQA, il existe une marge d'amélioration. Bien sûr, on voudrait dans l'idéal utiliser le moins de données annotées possible pour combler cet écart car plus on utilise de telles données annotées manuellement plus on diminue l'intérêt d'utiliser un modèle général sur cette tâche. La suite de cette section présente donc une méthode de génération de questions non-supervisée permettant d'affiner des modèles généraux tels que UnifiedQA sur de nouveaux domaines sans recours à des annotateurs humains et donc sans le coût en temps et en mains d'œuvre qu'une telle annotation demande.

4.4.1. Bases de données utilisées

Pour cette expérience, nous avons besoin d'une base de données n'ayant pas été vue par UnifiedQA lors de son entraînement afin de démontrer l'intérêt de notre méthode sur de nouveaux domaines. A cet effet, j'ai sélectionné la base de données SciQ (Welbl, N. F. Liu et Gardner 2017). SciQ est une base de données constituée de questions à choix multiples avec 4 choix de réponse possible, très similaire dans sa forme à OpenBookQA ou ARC. Les domaines abordés sont la physique, la chimie et la biologie. Cependant, contrairement à OpenBookQA et ARC qui ont été utilisés pour l'apprentissage de UnifiedQA, SciQ contient des questions sur des sujets beaucoup plus pointus et précis. De plus, SciQ n'a pas été vu par UnifiedQA lors de son entraînement. La figure 19 présente un exemple de question dans la base de données SciQ. De plus, mes expériences réalisées sur cette base de données ont montré qu'un affinage de UnifiedQA sur les données d'entraînement spécifiques à SciQ produit de bien meilleurs résultats qu'une application directe sans affinage. Mon objectif avec les expériences qui suivent est de montrer qu'il est possible de réduire une partie de cet écart grâce à la méthode de génération non-supervisée présentée ci-après. Dans le but d'asseoir les

Question :

What type of organism is commonly used in preparation of foods such as cheese and yogurt ?

(A) mesophilic organisms (B) protozoa

(C) gymnosperms (D) viruses

Support text :

Mesophiles are often found living in or on the bodies of humans or other animals. The optimal growth temperature of many pathogenic mesophiles is 37 (98), the normal human body temperature. Mesophilic organisms have important uses in food preparation, including cheese, yogurt, beer and wine.

Figure 19 – Un example de question dans SciQ. La bonne réponse est indiquée en gras.

résultats, j'ai également testé mon approche sur deux autres bases de données de question-réponse à choix multiples : CommonSenseQA (Talmor et al. 2019) et QASC (Khot et al. 2020), également absentes de l'entraînement d'UnifiedQA.

4.4.2. Processus de génération

La méthode proposée est en réalité une amélioration de la méthode présentée dans la section précédente avec un nouveau système de conversion des phrases en question. Le processus de génération de question est composé de trois parties présentées ci-après : La sélection des phrases, la transformation de ces phrases en questions et enfin le raffinage des distracteurs.

Le système de génération utilise un ensemble de phrases non annotées comme base pour générer les questions. Dans cette partie, je compare 3 sélections plus ou moins proches de la base de données de test et variant dans le niveau de supervision requis. Premièrement, je considère un scénario pour lequel aucune supervision n'est requise et où l'utilisateur se contente de donner le nom d'un ou plusieurs domaines d'intérêt. Dans notre cas les domaines que j'utilise sont "Physics", "Chemistry" et "Biology" qui sont les principaux domaines abordés dans SciQ. J'applique ensuite un algorithme d'extraction de données simple sur la base de données Wikipédia. L'algorithme génère une liste de catégories en explorant récursivement toutes les sous-catégories en partant des trois catégories principales énoncées précédemment. La profondeur de récursion est limitée à 4 afin d'éviter d'obtenir des thèmes trop éloignés

	Sentences	Questions
SciQ data	53 270	77 873
SciQ data (train only)	45 526	66 552
Wikipédia data	45 327	62 848

Tableau 11 – Nombre de phrases collectées dans chaque ensemble ainsi que le nombre de questions qui ont pu être générées à partir de ces phrases.

des trois catégories principales. Ensuite, les premiers paragraphes (càd la partie *résumé* se trouvant en haut de chaque page Wikipédia) de chacune des pages correspondant à ces catégories sont extraits. J'applique de plus un filtrage pour ne garder que les articles ayant plus de 800 visiteurs par jour en moyenne sur les dix derniers jours (extraction effectuée le 27 avril 2021) afin d'éliminer les pages traitant de sujets trop spécifiques. Cette sélection résulte en une base de données de 12 656 pages Wikipédia.

De plus, il est possible de penser que dans une situation réelle, l'utilisateur puisse avoir plus d'information sur le domaine d'intérêt et qu'il soit donc possible d'effectuer une sélection plus précise des phrases utilisées voir même que l'on dispose directement d'un ensemble de texte décrivant le domaine ou les thèmes sur lequel on souhaite entraîner notre modèle de question-réponse (par exemple un manuel scolaire ou une documentation technique). Dans cette optique, je compare la méthode de sélection décrite précédemment avec deux autres bases de données de phrases extraites directement de SciQ. Ces deux ensembles résultent donc en un apprentissage qui n'est pas entièrement non supervisé mais qui, je le pense, peut correspondre à un cas d'usage plausible de ce genre de méthode. La base de données SciQ fournit avec chaque question un court paragraphe donnant a priori l'information nécessaire pour répondre à la question. Quand on les rassemble et que l'on mélange les phrases, ces paragraphes forment un large ensemble de texte qui contient de l'information relative au domaine étudié sans pour autant être lié à une question en particulier. Je compare deux bases de données distinctes. Dans la première sont inclues les paragraphes associés aux questions des parties entraînement, développement et test de SciQ. Cette base de données simule une situation où la sélection des phrases est "parfaite" et où l'ensemble de phrases résultant contient toute l'information requise pour répondre aux questions se trouvant dans la partie test de SciQ. Toutefois, il est utile de se rappeler que j'utilise uniquement les paragraphes associés aux questions et non les questions annotées elles même. De plus la

Question :

What often is found living in or on the bodies of humans or other animals ?

Right answer : mesophile

Random distractors :

(A) the most magnetic material in nature

(B) this energy

(C) climate

Refined distractors :

(A) carbohydrates

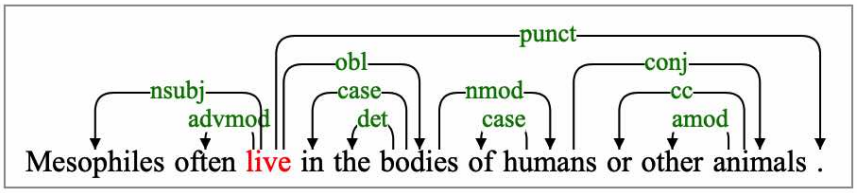
(B) small cell fragments called platelet

(C) echinoderm

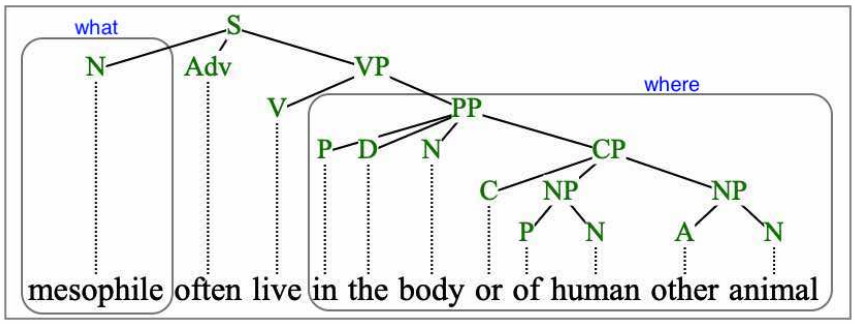
Figure 20 – Exemple de question synthétique générée à partir de la deuxième phrase du paragraphe support de la figure 19 avec un ensemble de distracteurs aléatoires et avec l'ensemble de distracteurs raffinés.

base de données générée par ce procédé ne conserve pas l'information de quelle phrase était associée à quelle question initialement. D'un point de vue supervision, cette méthode retire la charge d'annotation des questions d'entraînement mais demande cependant à l'utilisateur de rassembler un large ensemble de phrases couvrant les domaines pour lesquels il souhaite entraîner un système de question-réponse. Je compare cette première sélection avec une sélection où seuls les paragraphes issus de la partie entraînement sont inclus dans la sélection. Cette deuxième sélection représente un scénario plus réaliste dans le sens où l'utilisateur n'a pas besoin d'une sélection "parfaite" de phrases mais seulement de phrases proches du domaine d'intérêt. Le nombre de phrases extraites pour chacune de ces bases de données est présenté en Tableau 11.

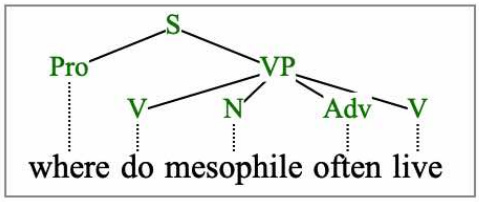
La génération de questions à partir d'une phrase s'appuie sur jsRealB (Lapalme 2021) qui génère une phrase affirmative à partir d'une structure syntaxique. Il peut également être paramétré pour générer des variations de la phrase originale telles que sa négation, sa forme passive et différents types de questions telles que *who*, *what*, *when*, etc. La structure syntaxique d'une phrase est le plus souvent créée par un utilisateur ou par un programme à



(a) Structure de dépendance



(b) Arbre Syntaxique



(c) Question générée

Figure 21 – Processus de génération de questions pour une phrase similaire à celle utilisée pour produire la question de la figure 20. Une analyse de dépendance (a) produite par Stanza est transformée en un arbre syntaxique (b) dans lequel deux sous-arbres peuvent être identifiés comme les réponses à deux questions : *What* et *Where*. (c) montre l'arbre syntaxique transformé pour la question *Where*.

partir de données. Dans ce travail, elle est construite à partir d'une structure en dépendances universelles (UD) en utilisant une technique développée pour *SR'19* (Lapalme 2019). La structure UD d'une phrase est le résultat d'une analyse en dépendances avec Stanza (Qi et al. 2020). Nous avons donc un pipeline composé d'un analyseur de dépendance neuronale, suivi d'un programme pour créer une structure syntaxique utilisée comme entrée pour un réalisateur de texte, tous deux en JavaScript.

Afin de créer des questions à partir d'une seule structure syntaxique, jsRealB utilise les transformations grammaticales *classiques* : pour une question *who*, il supprime le sujet (c'est-à-dire le premier syntagme nominal avant le syntagme verbal), pour une question *what*, il supprime le sujet ou l'objet direct (c'est-à-dire le premier syntagme nominal dans le syntagme

verbal) ; pour les autres types de questions (*when*,*where*), il supprime la première phrase prépositionnelle dans la phrase verbale. Selon la préposition, la question sera une question *when* ou une question *where*. Notez que la partie de la phrase que l'on enlève devient la réponse à la question. Un exemple de génération peut être trouvé en figure 21.

Afin de déterminer quelles questions sont appropriées pour une phrase donnée, la structure de dépendance de la phrase originale est examinée pour vérifier si elle contient la partie requise à supprimer avant de paramétrer la génération. Le tableau 11 indique le nombre de questions générées pour chaque ensemble de données. Un exemple de question synthétique est montré dans la figure 20.

Enfin, similairement à la section précédente, nous comparons deux méthodes de génération des distracteurs. L'une où je sélectionne les distracteurs pour une question donnée aléatoirement parmi les bonnes réponses à des questions de même type (ex : les distracteurs pour une question de type *where* seront sélectionnés parmi les réponses à d'autres questions de type *where* dans notre base de données de questions synthétiques). Et l'autre où j'utilise la méthode présentée en section 4.3 pour resélectionner les distracteurs. Cette fois j'utilise $N = 64$ candidats distracteurs pour chaque question. De manière générale, donner un plus grand nombre de candidats au système de raffinage des distracteurs semble toujours bénéfique tant qu'on garde ce nombre petit (en ordre de grandeur) par rapport à la quantité de questions. Pour mes expériences, le nombre de candidats est donc limité par le temps nécessaire pour scorer toutes les paires question/candidat avec RoBERTa.

4.4.3. Détails d'implémentation

Pour le raffinage des distracteurs, j'utilise un modèle RoBERTa "Large" affiné pour 4 époques avec un taux d'apprentissage de 1×10^{-5}. Ces hyperparamètres sont choisis en se basant sur mes expériences précédentes avec RoBERTa sur les bases de données OpenBookQA et ARC. L'affinage final sur UnifiedQA est fait en utilisant les mêmes hyperparamètres que ceux utilisés dans l'article original d'UnifiedQA (Khashabi, Min et al. 2020). Nous utilisons la version "Large" d'UnifiedQA affiné pendant 4 époques avec Adafactor et un taux d'apprentissage de 1×10^{-5}. Le taux d'apprentissage est choisi pour obtenir les meilleures performances lors de l'apprentissage supervisé d'UnifiedQA et les mêmes hyperparamètres sont utilisés pour tout mes modèles. Nous utilisons la bibliothèque HuggingFace

	Dev	**Test**
UnifiedQA (no fine-tuning)	64.6	63.4
UnifiedQA (supervised)	78.7	78.7
Unsupervised - Random distractors		
SciQ data	71.3	70.8
SciQ data (train only)	70.9	70.1
Wikipédia data	68.3	67.5
Unsupervised - Refined distractors		
SciQ data	75.4	74.2
SciQ data (train only)	73.1	72.4
Wikipédia data	70.6	69.4

Tableau 12 – Résultats sur SciQ de UnifiedQA affiné sur mes différentes bases de données synthétiques. “SciQ data” fait référence aux questions générées à partir des paragraphes support présents dans SciQ. “Wikipédia data” fait référence aux questions générées à partir de Wikipédia. Tous les scores sont des moyennes sur 3 apprentissages différents (incluant le processus complet de génération de questions et l’affinage de UnifiedQA).

tranformers (Wolf et al. 2020) pour l’implémentation et la récupération des modèles état de l’art.

4.4.4. Résultats

Les résultats présentés dans le tableau 12 ont un intervalle de confiance de Wald à 95% de ±2.8% et correspondent aux résultats obtenus par mes différents modèles sur SciQ. La première ligne du tableau 12 présente les résultats obtenus par un UnifiedQA sans affinage et la deuxième ceux obtenues par UnifiedQA affiné sur la base de données d’entraînement de SciQ. L’objectif est donc d’obtenir des résultats les plus proches possible d’un UnifiedQA supervisé en utilisant uniquement des méthodes non-supervisées pour l’affinage. Les résultats obtenus avec Wikipédia sont les seuls qui sont vraiment non-supervisés et donc sont les seuls directement comparables avec UnifiedQA sans affinage ou avec d’autres méthodes non-supervisées. Les autres résultats montrent ce qui peut être obtenu avec une meilleure sélection de phrases.

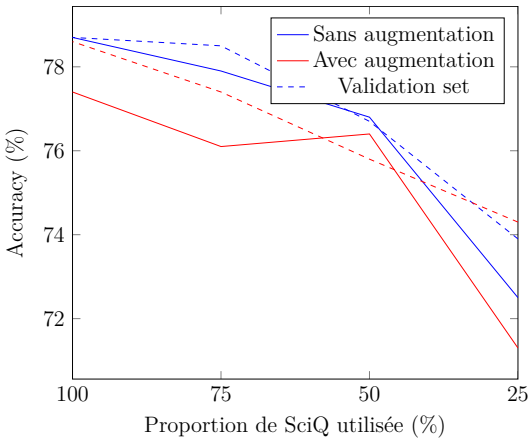
	CQA	**QASC**
UnifiedQA (no fine-tuning)	60.9	44.5
UnifiedQA (supervised)	74.3	61.0
Wikipédia data (Random)	64.9	57.2
Wikipédia data (Refined)	65.1	59.4

Tableau 13 – Résultats obtenus sur l'ensemble de développement de QASC et CommonSenseQA par un modèle UnifiedQA affiné sur les données issues de Wikipédia.

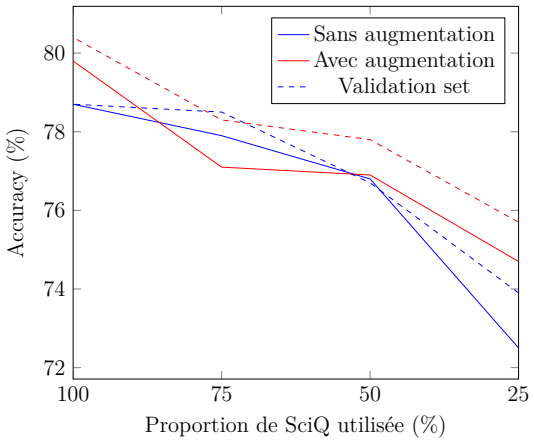
Une première observation est qu'un affinage de UnifiedQA avec des distracteurs aléatoires améliore déjà beaucoup les résultats obtenus par celui-ci comparé à ce même modèle sans affinage. De plus, comme attendu, plus les phrases sélectionnées sont proches des questions, plus les résultats sont bons. Ainsi, les meilleurs résultats sont obtenus dans la situation où on inclut les phrases issues de SciQ avec des résultats encore accrus quand on inclut les phrases issues de l'ensemble de test. Enfin, les questions extraites de Wikipédia augmentent aussi les résultats même si c'est dans une moindre proportion. Le raffinage des distracteurs permet de plus d'augmenter encore les résultats dans toutes les situations présentées. Cela semble indiquer qu'une meilleure sélection des distracteurs pour obtenir des questions plus difficiles est utile dans ce contexte et que la méthode de sélection semble adéquate au moins dans certaines situations.

Les résultats obtenus par notre méthode sur CommonSenseQA et QASC sont présentés en tableau 13. De manière générale, ces résultats confirment les conclusions présentées précédemment, à savoir une forte augmentation des performances en entrainant avec des distracteurs aléatoires et une augmentation supplémentaire en entrainant avec des distracteurs raffinés. On remarque cependant une différence moins importante entre les distracteurs aléatoires et raffinés. Cette plus faible différence peut être expliquée par le fait que les distracteurs dans les ensembles de développement de QASC et CommonSenseQA sont déjà plus simples que ceux de SciQ et que par conséquent faire un entraînement avec des distracteurs plus difficiles offre moins d'avantages.

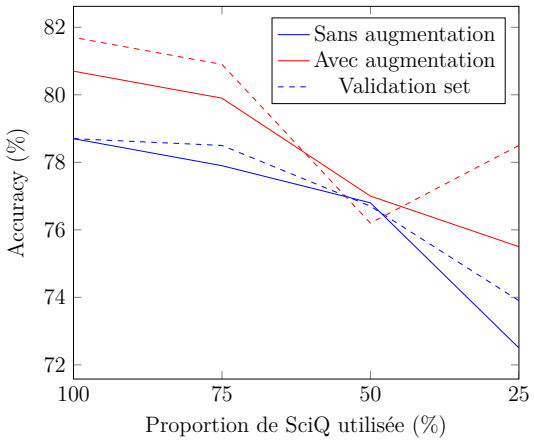
Finalement, il est possible d'imaginer que dans certains cas, il existe tout de même un petit nombre de données disponibles pour une tâche donnée. Dans cette optique, il serait intéressant d'utiliser les questions générées en parallèle de ces données déjà existantes en tant qu'augmentation de données. La figure 22 présente les résultats obtenus par une telle



(a) Augmentation avec les données synthétiques issues de Wikipédia



(b) Augmentation avec les données synthétiques issues de SciQ (données d'entraînement uniquement)

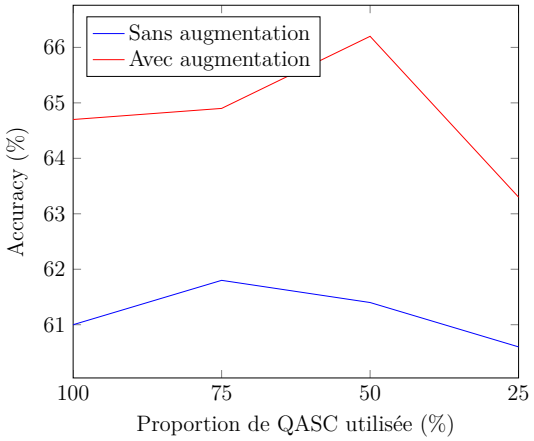


(c) Augmentation avec les données synthétiques issues de SciQ (données d'enrainement et de test)

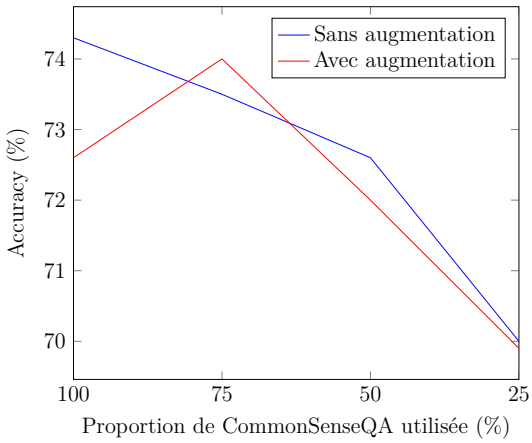
		Sans	**Wikipédia**	**SciQ (train)**	**SciQ (test + train)**
100%	dev	78.7	78.6	80.4	**81.7**
	test	78.7	77.4	79.8	**80.7**
75%	dev	78.5	77.4	78.3	**80.9**
	test	77.9	76.1	77.1	**79.9**
50%	dev	76.7	75.8	**77.8**	76.2
	test	76.8	76.4	76.9	**77.0**
25%	dev	73.9	74.3	75.7	**78.5**
	test	72.5	71.3	74.7	**75.5**

(d) Tableau récapitulatif des scores d'accuracy (%). La première colonne indique la proportion de l'ensemble d'entraînement de SciQ qui a été utilisée et la deuxième colonne indique sur quel ensemble (développement ou test) on effectue l'évaluation.

Figure 22 – Cette figure montre les résultats d'accuracy obtenues par différents modèles entraînés avec différentes proportions de la base de données SciQ ainsi que différentes bases de données utilisées comme augmentation (parmi les données issues de Wikipédia, SciQ et SciQ sans le jeu de test).



(a) Augmentation sur QASC avec les données synthétiques issues de Wikipédia



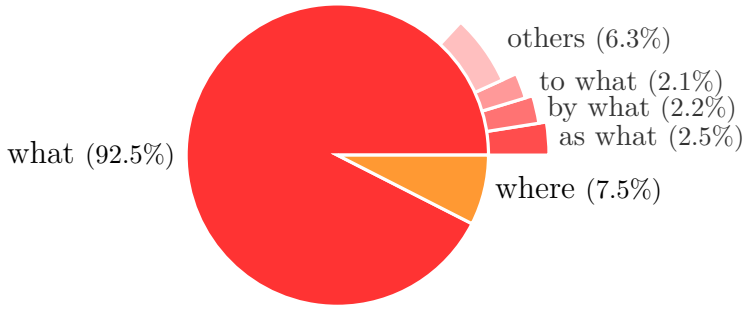
(b) Augmentation sur CommonSenseQA avec les données synthétiques issues de Wikipédia

	QASC		**CSQA**	
	Sans	Avec	Sans	Avec
100%	61.0	**64.7**	**74.3**	72.6
75%	61.8	**64.9**	73.5	**74.0**
50%	61.4	**66.2**	**72.6**	72.0
25%	60.6	**63.3**	**70.0**	69.9

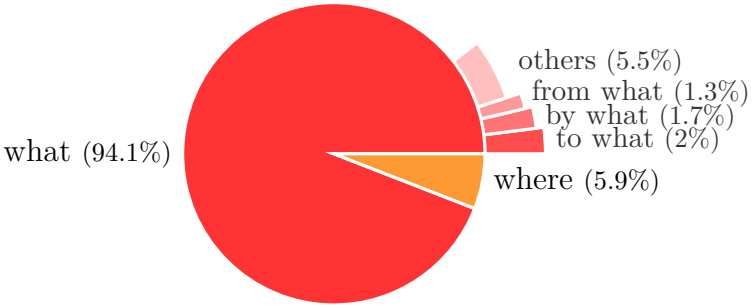
(c) Tableau récapitulatif des scores d'accuracy (%) pour QASC et CommonSenseQA avec ou sans augmentation de données. La première colonne indique la proportion du set d'entraînement de QASC ou CommonSenseQA qui a été utilisée.

Figure 23 – Cette figure montre les résultats d'accuracy obtenues par différents modèles entraîné avec différentes proportions des bases de données QASC et CommonSenseQA avec ou sans augmentation de données. Les données utilisées pour l'augmentation sont les questions générées à partir de Wikipédia.

augmentation. Pour ce faire les données générées sont mélangées aux données réelles et le modèle UnifiedQA est entraîné avec ces données similairement aux expériences précédentes. Je compare les résultats de l'augmentation de données avec différentes quantités de données réelles (25%, 50%, 75% ou 100%). On peut observer que l'augmentation de performances dépend beaucoup de la source des données synthétiques : là où l'utilisation des données issues de Wikipédia et de l'ensemble d'entraînement de SciQ ne semble pas apporter d'augmentation significative des performances par rapport à une utilisation des données réelles sans augmentation (avec même une légère diminution pour les données de Wikipédia), l'ajout de



(a) Données de Wikipédia



(b) Données de SciQ

Figure 24 – Répartition des types de questions dans les bases de données de questions générées à partir de SciQ et Wikipédia.

questions générées avec l'ensemble de test de SciQ permet une augmentation consistante des résultats obtenus. Cela montre qu'en matière d'augmentation de données, similairement aux résultats non-supervisés, il est possible d'améliorer les résultats obtenus par UnifiedQA avec nos données synthétiques du moment que l'on dispose d'une source de données suffisamment proche du dataset cible. De plus, comme pour la partie non-supervisée, j'ai répliqué les mêmes expériences sur CommonSenseQA et QASC (figure 23) en utilisant les questions générées à partir de Wikipédia. Ici, les résultats sont variables puisque si on n'observe aucune différence flagrante sur CommonSenseQA entre un entraînement avec ou sans augmentation, il semble que cette augmentation apporte beaucoup à UnifiedQA sur QASC. Il semble donc que l'efficacité de l'augmentation de données dépende aussi beaucoup du dataset cible sur lequel elle est appliquée.

4.4.5. Analyse qualitative

Je m'intéresse tout d'abord à la distribution des différents types de questions (*what*, *where*, *when*, etc.) présents dans les bases de données générées et réelle. La figure 24 présente la distribution des types de questions dans mes bases de données synthétiques (avec (a)

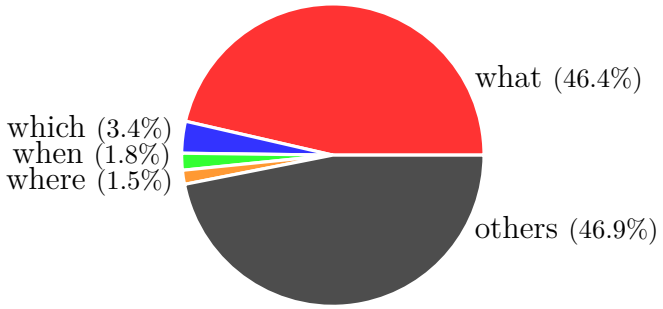


Figure 25 – Répartition des types de questions dans les ensembles d'entraînement, de developpement et de test de SciQ. La partie grisée correspond à des questions qui ne commencent pas directement par un mot interrogatif en *wh*.

les phrases issues de Wikipédia et (b) les phrases issues de SciQ) et la figure 25 présente la distribution réelle des questions dans la base de données SciQ. Une des premières observations que l'on peut faire est que mes bases de données synthétiques ne contiennent aucune question utilisant le mot interrogatif *when*. En effet, même si le système de génération des questions peut en théorie générer de telles questions, les conditions nécessaires pour cette génération sont pour l'heure trop restrictives pour avoir une chance significative d'obtenir ce type de questions dans le dataset final. Cependant, ce manque de questions temporelles ne semble pas être un problème majeur quand on regarde la distribution réelle des types de questions dans SciQ. On remarque que dans SciQ, les questions commençant par un mot interrogatif *what* sont extrêmement majoritaires. De plus, le deuxième mot interrogatif le plus fréquent est *which* qui, dans le contexte de questions à choix multiples, est souvent équivalent à *what* et les deux sont interchangeable dans un grand nombre de cas (ex : *"What is found living in or on the bodies of humans or other animals ?"* → *"Which is found living in or on the bodies of humans or other animals ?"*).

Une autre interrogation majeure sur mon processus de génération concerne la présence potentielle de distracteurs qui sont des bonnes réponses pour la question considérée. Bien que les réseaux de neurones soient plutôt résistants à la présence de quelques mauvais exemples d'entraînement, il est important de vérifier que le processus de génération des questions et en particulier le processus de raffinage des distracteurs ne produit pas trop de questions avec plusieurs bonnes réponses au risque de dégrader l'entraînement des modèles. Pour étudier ce phénomène, j'ai utilisé une annotation manuelle humaine. Dans un premier temps, j'ai considéré faire annoter l'une de mes bases de données de questions construites à partir de

SciQ. Cependant, les questions dans SciQ portent sur des domaines scientifiques plutôt pointus et il est donc apparu qu'il serait difficile pour des annotateurs sans aucune connaissance des domaines abordés d'identifier correctement et rapidement les réponses aux questions. J'ai donc décidé d'utiliser OpenBookQA à la place. OpenBookQA contient des questions plus générales et d'un niveau plus abordable et donc une annotation sur ce jeu de données permet d'obtenir des résultats moins bruités par le niveau de connaissance des différents annotateurs.

A partir des questions d'OpenBookQA (ensemble d'entraînement) je crée 3 bases de données distinctes contenant les mêmes questions mais avec des distracteurs différents. Le premier ensemble de questions correspond aux questions telles qu'elles sont présentes dans OpenBookQA avec leurs distracteurs originaux. Le deuxième ensemble contient les questions d'OpenBookQA mais avec des distracteurs choisis aléatoirement parmi les bonnes réponses aux autres questions dans OpenBookQA. Enfin, le troisième ensemble contient des questions associées à des distracteurs raffinés obtenus en utilisant ma méthode de raffinage des distracteurs. J'ai demandé ensuite à 3 annotateurs de répondre aux questions en choisissant zéro, une ou plusieurs réponses parmi les choix de réponses associés à la question. Dans la mesure du possible, je m'arrange pour que la plupart des annotateurs soient ignorants du fait que les questions du dataset initial n'attendent qu'une seule réponse. De plus, lors du processus d'annotation, je mélange les questions issues des trois ensembles (distracteurs originaux, distracteurs aléatoires et distracteurs raffinés) et les questions sont présentées aux annotateurs sans aucune indication de leur origine. 600 questions sont annotées de cette manière.

Les résultats de cette annotation sont présentés en table 14. Je commence par analyser la proportion de questions pour lesquelles les annotateurs ont choisi plusieurs réponses. Je me limite aux questions pour lesquelles le choix de réponse correct a été sélectionné par les annotateurs. En effet, si ce n'est pas le cas, on peut supposer que l'annotateur n'a pas compris la question ou qu'il ne disposait pas de la connaissance requise pour répondre. La proportion de questions avec plusieurs bonnes réponses dans mes trois datasets (*orginal*, *aléatoire* et *raffiné*) est donc respectivement de 11.5%, 6.7% et 21%. Comme on pouvait s'y attendre, les questions avec des distracteurs aléatoires ont un plus faible pourcentage de questions avec plusieurs bonnes réponses car il est très improbable de tomber au hasard sur un distracteur qui est une réponse correcte. On observe aussi que le dataset avec des distracteurs raffinés

		Annotateur 1	Annotateur 2	Annotateur 3	Total
Original	Nbre de Questions	68	65	56	189
	Réponses Correctes	58	58	40	156
	Réponses Correctes (Strictement)	49	53	36	138
	Questions avec Plusieurs Réponses	9	5	4	18
Aléatoire	Nbre de Questions	63	75	69	207
	Réponses Correctes	59	73	48	180
	Réponses Correctes (Strictement)	53	71	44	168
	Questions avec Plusieurs Réponses	6	2	4	12
Raffiné	Nbre de Questions	69	60	75	204
	Réponses Correctes	51	49	52	152
	Réponses Correctes (Strictement)	44	36	40	120
	Questions avec Plusieurs Réponses	7	13	12	32

Tableau 14 – Récapitulatif des résultats de l'annotation. Les réponses correctes (les questions pour lesquelles l'annotateur a au moins inclut la bonne réponse parmi ces choix) sont séparées en 2 catégories : d'une part les réponses strictement correctes c'est à dire les questions pour lesquelles l'annotateur a choisi la bonne réponse et uniquement celle-là et celles pour lesquelles l'annotateur a choisi plusieurs réponses.

contient presque 2 fois plus de questions avec plusieurs bonnes réponses. Ce chiffre n'est pas forcément problématique pour l'apprentissage de nos modèles mais il reste significatif et doit donc être pris en compte lorsque l'on utilise cette méthode de sélection des distracteurs.

On peut également noter que les questions avec des distracteurs raffinés semblent plus difficiles. La proportion de questions pour lesquelles l'annotateur a au moins inclus la bonne réponse parmi ces choix est de 83.5%, 87% et 74.5% pour les ensembles *original*, *aléatoire* et *raffiné* respectivement. Le fait que les questions avec des distracteurs aléatoires soient plus faciles est plutôt normal et attendu. En revanche, il est surprenant que les questions avec des distracteurs raffinés soient plus difficiles comparées aux questions originales. Une explication possible à ce phénomène est que, malgré le fait que le processus de raffinage des distracteurs

ne modifie pas les questions elles-mêmes (ni les bonnes réponses), il est tout de même possible que certaines questions n'aient plus de sens une fois la resélection des distracteurs effectuée. C'est le cas par exemple lorsque la réponse attendue à la question est de la forme "None of these" ou bien "All of them". Le dataset OpenBookQA sur lequel est effectué l'annotation est particulièrement concerné puisqu'il contient beaucoup de ce type de questions.

4.4.6. UnifiedQA-v2

Depuis mes premières expériences, une nouvelle version UnifiedQA-v2 (Khashabi, Kordi et Hajishirzi 2022) est disponible. J'ai donc reproduit quelques-unes de mes expériences sur ce nouveau modèle afin de voir comment mes conclusions évoluent avec un modèle supposément plus puissant. La principale différence entre UnifiedQA et cette nouvelle version UnifiedQA-v2 est que cette dernière a été entraînée en utilisant plus de bases de données. L'entraînement d'UnifiedQA-v2 utilise par exemple désormais les ensembles CommonSenseQA et QASC. Il n'est donc plus possible d'utiliser ces deux bases de données pour tester notre méthode non supervisée. Les expériences reportées ici seront donc uniquement sur SciQ. Je reprends donc les expériences dont les résultats sont présentés dans le tableau 12 avec le même protocole expérimental et les nouveaux résultats obtenus en utilisant UnifiedQA-v2 sont présentés dans le tableau 15. La principale différence est que l'utilisation de questions produites à partir des données provenant de Wikipédia n'apporte plus d'amélioration des résultats d'UnifiedQA-v2 sur SciQ. Ceci montre que cette version améliorée du modèle qui généralise mieux à de nouvelles bases de données ne bénéficie plus des données trop vaguement sélectionnées dans Wikipédia. En revanche, il est toujours possible d'obtenir de meilleures performances en utilisant des données plus proches du dataset cible (comme des questions générées à partir des phrases de contexte de SciQ).

4.5. Génération de questions par *denoising autoencoder*

Au début des expériences réalisées dans ce chapitre, j'ai commencé par expérimenter avec différentes méthodes de génération non-supervisée de questions. Ma première approche était l'application de "*denoising autoencoders*" (Vincent et al. 2008). L'idée avec ce modèle est de reconstituer un exemple à partir d'entrées bruitées ou incomplètes. Bien que cette méthode

	Dev	**Test**
UnifiedQA-v2 (no fine-tuning)	71.9	67.3
UnifiedQA-v2 (supervised)	80.9	79.1
Unsupervised - Random distractors		
SciQ data	72.8	72.0
SciQ data (train only)	72.3	70.7
Wikipédia data	69.6	67.5
Unsupervised - Refined distractors		
SciQ data	77.6	75.6
SciQ data (train only)	75.8	75.7
Wikipédia data	71.0	69.5

Tableau 15 – Résultats sur SciQ de UnifiedQA-v2 affiné sur mes différentes bases de données synthétiques. Cette table reprend les expériences reportées en table 12 en remplaçant UnifiedQA par UnifiedQA-v2.

n'ait pas été retenue pour la suite, il m'a semblé utile de la mentionner ici puisqu'elle produit néanmoins des résultats prometteurs qui pourraient faire l'objet de futures recherches.

L'idée avec ce modèle est de reconstituer un exemple (dans notre cas une question) à partir d'entrées bruitées ou incomplètes. A la manière d'un auto-encodeur classique, le réseau est tout d'abord constitué d'une partie encodeur (comme par exemple un réseau LSTM (Hochreiter et Schmidhuber 1997a) ou plus récemment un "*transformer*" (Vaswani et al. 2017)) qui a pour rôle de compresser l'information reçue en entrée dans un vecteur de représentation de taille fixe. Ensuite, une partie décodeur avec une architecture similaire à l'encodeur reconstruit l'entrée initiale du réseau à partir de ce vecteur de représentation. Dans un "*denoising autoencoder*", l'entrée du réseau est bruitée et l'objectif pour ce réseau est de reconstituer l'entrée initiale non bruitée. La plupart du temps, l'entraînement de ce réseau se fait de manière auto-supervisée. C'est à dire qu'à partir d'un ensemble de données non-annotées, on bruite les données automatiquement (pour du texte, on peut mélanger les mots entre eux ou bien en supprimer certains) pour produire les couples entrées/sorties qui serviront à l'entraînement.

L'idée derrière l'application de cette méthode à la génération de questions est la suivante : certaines des méthodes déjà utilisées dans l'état de l'art pour la génération non-supervisée

de questions grâce à des méthodes d'apprentissage profond reposent sur des méthodes assez complexes s'inspirant de la traduction automatique non-supervisée (Lewis, Denoyer et Riedel 2019). Pour la traduction, ces méthodes se justifient puisqu'il n'y a a priori pas nécessairement de liens (autres que sémantiques) entre l'entrée dans une langue et la phrase traduite de l'autre côté. Si les langues en question sont suffisamment distantes, les mots et les structures grammaticales n'ont aucune raison d'être les mêmes, ce qui nécessite ces méthodes particulières. Dans le cas de la génération de questions à partir de phrases affirmatives dans lesquelles on a choisi une réponse, la situation est différente. En effet, la plupart du temps, la majorité des mots (hors *stopwords*) dans la phrase initiale sont aussi ceux que l'on retrouve dans la question et la tâche d'écriture d'une question revient donc à réorganiser les mots. Ainsi, la phrase "*Plants are producers.*" devient "*Plants are* ____." puis "*What are plants*?" lorsqu'on extrait "*producers*" comme réponse correcte et qu'on reformule ce qui reste. La procedure d'entraînement serait donc la suivante : à partir d'un ensemble non-annoté de questions (c'est-à-dire sans les réponses associées), il s'agit pour chaque question de retirer le mot interrogatif qu'elle contient, de potentiellement mélanger les mots, de supprimer les *stopwords* et finalement de supprimer aléatoirement certains mots. On entraîne ensuite un réseau texte à texte (peut-être un réseau pré-entraîné comme T5) à réécrire la question à partir de cet ensemble de mots bruités. Au moment de la génération de questions, on utilise un ensemble non-annoté de phrases du domaine sur lequel on veut être capable de répondre. A partir de ces phrases, on sélectionne un morceau du texte au hasard pour jouer le rôle de réponse (par exemple une phrase nominale ou verbale) et on applique au reste des mots dans la phrase un bruitage similaire à celui appliqué lors de l'entraînement. En donnant ensuite ces entrées à notre modèle, on récupère en sortie des questions, qu'on peut joindre aux parties de texte qui ont été retirées pour former des couples d'entraînement pour un modèle de question-réponse. Cette approche utilisant un "*denoising autoencoders*" a déjà été appliquée avec succès à la génération de texte comme dans (Freitag et Roy 2018).

Pour appliquer cette méthode, j'ai besoin d'un ensemble non-annoté de questions. Pour mes premières expériences, j'ai utilisé un ensemble de questions extraites de la base de données SQUAD. Ces questions sont donc extraites d'une base de données annotée car plus rapides à obtenir pour les besoins des expériences mais on peut imaginer extraire des questions du Web par exemple pour une utilisation en situation réelle. Une des conséquences

Entrées	**Sorties**
represent's center Where be the Holy Spirit France largely you're Huguenot population urban planning agency that cover the GHMC its suburbs, extend to be 54 mandals in should've five districts encircle the city	Where was France's Huguenot population largely centered ?
Armenians enjoy only be Swiss city What had the center of the Calvinist bear movement ma can amend	What Swiss city was the center of the Calvinist movement ?

Figure 26 – Exemples issues de la base de données d'entraînement générée pour le système de génération de questions par *denoising auto-encoder*.

néanmoins est que les questions générées sont des questions adaptées pour du question-réponse extractif du type de SQUAD. Afin de former la base de données d'entraînement, je bruite donc ces questions en utilisant une combinaison de plusieurs heuristiques :

- **Mélange par partie.** Je découpe la question d'entrée en plusieurs parties (jusqu'à 3). Ces parties sont alors mélangées. Cette méthode permet de simuler le réarrangement des mots lors de la conversion d'une phrase en question sans mélanger l'intégralité des mots dans la phrase.
- **Ajout et retrait de *stopwords*.** Je supprime et j'ajoute des *stopwords* aléatoires afin de donner plus de liberté aux modèles.
- **Ajout de parties aléatoires.** J'ajoute également des bouts de phrases aléatoires au début et à la fin de l'exemple généré pour simuler des questions portant sur seulement une partie de la phrase d'entrée.
- **Lemmatisation.** Je lemmatise les verbes et adjectifs afin de permettre aux modèles de changer la forme des verbes au besoin.

La figure 26 présente quelques exemples de couples entrée/sortie générés par cette méthode. Ces exemples d'entraînement sont ensuite utilisés pour entraîner un modèle T5 Large.

Au moment de générer les questions, on sélectionne un exemple de phrases (issues de Wikipédia par exemple) et on leur applique la transformation suivante. On commence par sélectionner un entité nommée aléatoire qui servira de réponse à la question générée. On remplace ensuite cette entité par un mot interrogatif correspondant à son type. On passe ensuite cette phrase modifiée à travers le modèle T5 entraîné à retrouver des questions à

Phrase d'entrée	**Question**
It's capital and largest city is Nairobi(What).	What is the capital and largest city ?
Construction differs from manufacturing in that manufacturing typically involves mass production(What) of similar items without a designated purchaser, while construction typically takes place on location for a known client.	What does manufacture typically involve ?
An Old Fashioned Boy is a surviving 1920 American silent comedy romance film directed by Jerome Storm and starring Charles Ray(Who).	Who starred in the 1920 American silent comedy film directed by Jerome Storm

Figure 27 – Exemples de bonnes questions générées par le système de *denoising auto-encoder*. Le mot souligné correspond à l'entité nommée choisie comme réponse et le mot entre parenthèses correspond au mot interrogatif par lequel elle est substituée.

Phrase d'entrée	**Question**
Kenya, officially the republic of Kenya, is a country in Africa and a founding member of the East African Community(What).	What is a founding member of the republic of Kenya ?
It is bordered by Tanzania(What) to the south, Uganda to the west, South Sudan to the north-west, Ethiopia to the north and Somalia to the north-east.	What is bordered by ?

Figure 28 – Exemples de mauvaises questions générées par le système de *denoising auto-encoder*. Le mot souligné correspond à l'entité nommée choisie comme réponse et le mot entre parenthèses correspond au mot interrogatif par lequel elle est substituée.

partir d'une version bruitée et on obtient en sortie une question. Cette méthode fonctionne très bien pour certains exemples comme illustré en figure 27 mais malheureusement, ce n'est pas le cas pour tous les exemples testés. Ainsi en figure 28 on peut voir que dans certains cas, la question ne correspond pas à l'entité choisie ou bien n'est pas bien formée.

Malgré des premiers résultats prometteurs, je n'ai pas retenu cette méthode de génération pour la suite de mes travaux et cela pour plusieurs raisons. Premièrement, même si certaines questions sont bonnes, il y a aussi beaucoup de questions incorrectes et mal formées qui pourraient impacter les résultats obtenus par un modèle entraîné sur celles-ci. De plus, cette

méthode nécessite l'entraînement d'un modèle T5 juste pour la génération des questions et cet entraînement demande beaucoup de temps. Enfin, le type des questions générées dépend de l'ensemble sur lequel on entraîne le modèle, ce qui demanderait de trouver un moyen de contraindre la récolte des questions afin d'obtenir uniquement des questions correspondant au type des questions dans la base de données cible. J'ai donc décidé de me tourner vers les méthodes de génération plus simples présentées au début de ce chapitre.

4.6. Conclusion

Dans ce chapitre, je me suis intéressé à la génération automatique non-supervisée de questions pour la tâche de question-réponse à choix multiples. Ce type de méthodes permet entre autres, en contrôlant la manière dont les questions sont générées de minimiser la présence des biais qui se retrouvent fréquemment dans les bases de données annotées humainement.

Dans un premier temps, j'ai commencé par utiliser des méthodes simples afin de générer des questions et leurs distracteurs associés et j'ai constaté qu'une telle méthode permet d'obtenir des résultats prometteurs lorsqu'appliqué à l'entraînement non-supervisé de modèles de question-réponse. Cependant, il est apparu qu'un entraînement non-supervisé ne parviendrait jamais au niveau de l'état de l'art supervisé. De plus, l'intérêt d'une approche non-supervisé s'estompe lorsqu'on considère le fait qu'il existe déjà des modèles généraux très performants pour le question-réponse.

En revanche, j'ai montré qu'il est possible grâce à ces nouvelles données générées automatiquement d'affiner ces modèles généraux pour leur permettre de mieux performer sur de nouvelles bases de données qu'ils n'ont pas vu pendant leur entraînement. Cet affinage permet donc d'adapter ces modèles à de nouveaux domaines sans coût d'annotation supplémentaire et en utilisant simplement un ensemble de phrases. Je montre que cet ensemble de phrases peut par exemple être extrait de Wikipédia donnant ainsi une bonne augmentation de performances sur UnifiedQA. Cependant, une sélection plus précise des phrases permet des performances supérieures et cette sélection s'avère même nécessaire pour obtenir un gain de performance sur des modèles plus récents comme UnifiedQA-v2.

Chapitre 5

Modèle multi-tâche pour mitiger les effets des biais

Dans ce chapitre, je montre qu'entraîner un modèle de reconnaissance optique de texte avec un vaste éventail de bases de données avec des styles et caractéristiques différents permet d'améliorer les résultats obtenus par ce modèle. Cette méthode qui est déjà largement utilisée sur d'autres tâches de TAL n'a, à ma connaissance, pas encore été testée sur l'OCR.

5.1. Introduction

Dans cette dernière partie j'explore une autre méthode de réduction de l'effet des biais. Pour la tâche de question-réponse, il est désormais assez courant de voir des modèles état de l'art entraînés sur plusieurs bases de données simultanément. Cette agrégation de données permet d'avoir plus de données d'entraînement dans un domaine où l'obtention de données annotées est difficile mais permet aussi de limiter l'impact des biais qui pourraient se trouver dans ces données. En effet, chaque base de données dans l'ensemble d'entraînement a des biais qui lui sont propres mais qui ne se retrouvent pas forcément dans les autres bases de données. Si le nombre de bases de données utilisées est suffisamment grand, alors ces biais particuliers à une base de données vont se retrouver noyés dans l'ensemble d'entraînement, ce qui permet au modèle de mieux identifier quelles sont les informations utiles à la tâche considérée. UnifiedQA (voir chapitre 4) est un exemple de ce type de modèle qui en plus

d'utiliser 8 bases de données pour son entraînement, est de plus entraîné sur plusieurs types de question-réponse différents (choix multiples, oui/non, extractif et abstractif).

Au cours de ma dernière année de doctorat, j'ai été financé par l'entreprise LexrockAI et j'ai travaillé avec eux sur un projet de Reconnaissance Optique de Caractères (OCR). J'ai travaillé en particulier sur la tâche de reconnaissance de mot où les modèles ont la charge de retranscrire un mot ou une suite de caractères à partir d'une image. En prenant inspiration sur ce qui se fait en question-réponse, j'ai étudié la possibilité d'entraîner un modèle d'OCR utilisant un grand nombre de bases de données différentes similairement à UnifiedQA pour le question-réponse. Pour la reconnaissance de mot, il existe plusieurs types de bases de données avec différentes caractéristiques : on différencie par exemple les bases de données d'écriture manuscrite (IAM (Marti et Bunke 1999)) de celles avec du texte tapuscript (DocBank (M. Li et al. 2020)) qui même si elles ont des particularités en commun (par exemple un texte qui est le plus souvent noir sur blanc) ne sont pas souvent utilisées ensemble pour entraîner des modèles. Un autre type de bases de données couramment utilisé correspond aux datasets de *Scene Text Recognition* (STR) comme SVT (K. Wang et Belongie 2010) constitués de textes dans des environnements plus complexes (enseignes de magasins par exemple). Ce type d'OCR se caractérisent par des textes possiblement en couleur, pas forcément écrits droits ou sur des fonds non unis. Malgré ces différences, il existe, comme pour le question-réponse, un grand nombre de points communs entre ces différents types de bases de données d'OCR. Par exemple, même si la forme des caractères varie entre différentes polices ou entre de l'écriture manuscrite et tapuscripte, il existe cependant certains points communs : beaucoup de gens ont une écriture manuscrite avec des caractères séparés les uns des autres qui se rapproche de certaines polices de caractères et inversement, certaines polices imitent l'écriture manuscrite. Un exemple de ce genre de rapprochement est par exemple le logo de "Coca Cola" qui pourrait se trouver dans certaines bases de données de *Scene Text Recognition* (STR) et pour lequel, à cause de la police utilisée, un modèle pourrait bénéficier d'un apprentissage sur une base de données d'écriture manuscrite. Il est donc vraissemblable que l'information apprise à partir d'un dataset soit transférable à un autre.

La plupart de ces bases de données d'OCR ont des biais assez flagrants. Par exemple, les textes présents dans le dataset IAM sont presque tous écrits en noir sur fond blanc, ce qui crée des performances dégradées si les textes dans l'ensemble de test n'ont pas la même

couleur. La police utilisée est aussi un biais fréquemment rencontré comme par exemple dans la base de données DocBank. Cette base de données est constituée de documents PDF extraits du site web arxiv.org. Comme les documents présents sur cette plateforme sont des articles scientifiques, ils suivent tous un format standard et sont écrits avec un ensemble très restreint de polices de caractères, ce qui entraîne une faible performance des modèles lorsque qu'on les applique à d'autres types de polices.

L'idée de ce chapitre est donc d'utiliser toutes ces bases de données pour entraîner un modèle unifié de reconnaissance de texte dans l'objectif de créer un modèle avec le moins de biais possible et suffisamment de pouvoir de généralisation pour être capable de bien performer sur de nouvelles bases de données. Dans ce chapitre je me concentre sur la tâche de transcription d'une image contenant un seul mot en texte et je laisse donc de côté la tâche connexe de détection de l'emplacement des mots dans une page. Les images qui sont données aux modèles sont donc déjà prédécoupées au besoin pour qu'elles ne contiennent qu'un seul mot.

5.2. Etat de l'art

Au moment du début de mes travaux sur l'OCR, l'état de l'art sur la tâche de *Scene Text Recognition* était constitué des modèles SATRN (Lee et al. 2020), SRN (Yu et al. 2020) et RCEED (Cui et al. 2021), tous trois atteignant un peu plus de 91% d'*accuracy* sur SVT (K. Wang et Belongie 2010). La plupart des modèles proposés au moment sur cette tâche utilisent les deux larges bases de données synthétiques SynthText (Gupta, Vedaldi et Zisserman 2016) et MJSynth (Jaderberg et al. 2014) pour l'entraînement même si quelques exemples notables comme RCEED utilisent en parallèle les ensembles d'entraînement de certaines bases de données annotées comme SVT (K. Wang et Belongie 2010), IIIT5K (Mishra, Alahari et Jawahar 2012) ou ICDAR13 (Karatzas, Shafait et al. 2013). Depuis, de plus en plus d'articles scientifiques ont montré l'intérêt d'utiliser des bases de données constituées de données réelles pendant l'entraînement des modèles en complément des données synthétiques et ce, même si ces données réelles sont en quantité très modeste comparées aux données synthétiques. Ainsi la plupart des modèles état de l'art actuels utilisent des données réelles dans leur entraînement (Loginov 2021 ; Y. He et al. 2022 ; Bautista et

(a) IAM **(b)** DocBank **(c)** SynthText **(d)** MJSynth

Figure 29 – Quelques exemples pour IAM, DocBank, SynthText et MJSynth.

Atienza 2022). Le modèle état de l'art actuel sur SVT [1] est PARSeq (Bautista et Atienza 2022). Sur d'autres tâches d'OCR comme la reconnaissance d'écriture manuscrite, il est plus difficile d'extraire un seul modèles état de l'art, du fait que la principale base de données utilisée, IAM (Marti et Bunke 1999), peut être utilisée de plusieurs manières. En effet, certains modèles utilisent le prédécoupage au niveaux des mots ou des lignes (Chaudhary et Bali 2022) alors que d'autres travaillent sur le paragraphe entier (Coquenet, Chatelain et Paquet 2022).

5.3. Datasets

5.3.1. Entraînement

Pour l'entraînement, j'ai choisi un ensemble de bases de données d'OCR avec différentes caractéristiques en essayant de couvrir au mieux l'ensemble des sous-tâches d'OCR. Ces bases de données (décrites ci-dessous) sont choisies pour leur taille et la qualité des données. L'ensemble d'entraînement comprend un ensemble de données réelles et synthétiques. Toutes les bases de données choisies contiennent du texte en anglais. L'idée ici est qu'un modèle sera plus performant à résoudre une tâche d'OCR s'il incorpore un modèle de langue lui permettant d'inférer certains caractères illisibles ou ambigus en fonction du contexte. Il est donc important d'avoir une uniformité dans les langues utilisées dans les différents datasets.

1. d'après le leaderboard de la base de données SVT.

IAM (Marti et Bunke 1999) : La base de données *"IAM Handwriting Dataset"* est une base de données constituée d'un ensemble de documents contenant de l'écriture manuscrite. Au total, la base de données contient 1 539 pages scannées. Plusieurs découpages sont fournis pour permettre d'extraire les lignes, les phrases ou les mots individuels. Pour les expériences qui vont suivre, j'utilise le découpage en mots d'IAM qui contient 115 320 examples d'entraînement. Ces mots sont séparés en 100 320 mots pour l'ensemble d'entraînement, 5 000 mots pour l'ensemble d'évaluation et 10 000 mots pour l'ensemble de test.

DocBank (M. Li et al. 2020) : DocBank est une base de données contenant 500 000 documents pdf extraits d'articles scientifiques. Les documents sont extraits à partir de la base de données d'arxiv.org. Une fois les documents découpés en mots, on obtient plus de 34 millions d'exemples. Comme cette base de données est bien plus grande que les autres bases de données utilisées, je n'utilise que les 5 premiers millions d'exemples pour l'ensemble d'entraînement. J'extrais également 5 000 et 10 000 exemples pour les ensembles de validation et de test respectivement.

SynthText (Gupta, Vedaldi et Zisserman 2016) : SynthText est une base de données synthétique constituée d'images dans lesquelles du texte a été automatiquement inséré. La base de données comprend un peu moins de 6 millions de mots dans environ 800 000 images.

MJSynth (Jaderberg et al. 2014) : Similairement à SynthText, MJSynth est une base de données synthétique couramment utilisée pour l'entraînement de modèles sur des bases de données d'OCR de type "Text in the wild". La base de données est constituée d'environ 9 millions d'images contenant un seul mot. Ces images sont générées automatiquement avec différentes polices et autres transformations aléatoires.

5.3.2. Evaluation

Pour l'évaluation des modèles, j'utilise un ensemble de données réelles. En plus de IAM et DocBank, j'évalue les modèles sur cinq autres bases de données : FUNSD, SVT, IIIT5K, ICDAR13 et ICDAR15. Ces cinq nouveaux datasets ne contiennent pas d'ensemble de validation et j'utilise donc leur ensemble d'entraînement comme ensemble de validation similairement à ce qui est fait dans d'autres travaux du domaine.

(a) FUNSD **(b)** SVT **(c)** IIIT5K

Figure 30 – Quelques exemples pour FUNSD, SVT et IIIT5K.

FUNSD (Jaume, Ekenel et Thiran 2019) : FUNSD est un dataset d'environ 200 formulaires scannés. Chaque document contient principalement du texte tapuscript mais également quelques exemples manuscripts comme des dates et des signatures. Comme cette base de données contient à la fois des mots et des phrases, je filtre et n'utilise que les exemples qui sont constitués d'un seul mot. Il en résulte un ensemble d'entraînement de 3 134 mots et un ensemble de test de 879 mots.

Street View Text (SVT) (K. Wang et Belongie 2010) : SVT est une base de données d'OCR de type "Text in the wild" contenant des images issues de Google Street View. Les images contiennent du texte sous la forme d'enseignes de magasin et de panneaux. Comme pour les autres bases de données, je travaille sur une version de la base où les parties d'intérêt de l'image (celles qui contiennent les mots) sont déjà prédécoupées. La base de données contient 257 et 647 mots pour les ensembles d'entraînement et de test respectivement.

IIIT5K (Mishra, Alahari et Jawahar 2012) : IIIT5K est un dataset similaire à SVT contenant des images issues de recherches dans Google Images. Les images inclues des panneaux, des numéros de maison, etc. IIIT5K contient 2000 et 3000 mots pour les ensembles d'entraînement et de test respectivement.

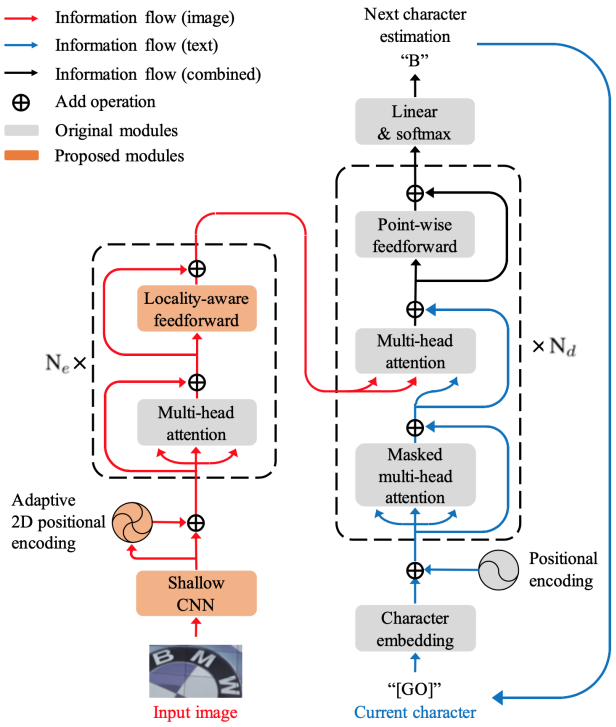


Figure 31 – Schéma de l'architecture du modèle SATRN. Image de Lee et al. (2020).

ICDAR13 (Karatzas, Shafait et al. 2013) : ICDAR13 est une autre base de données de type "*text in the wild*" très simillaire à SVT et IIIT5K. Elle contient un ensemble d'entraînement contenant 848 mots et un ensemble de test de 1095 mots.

ICDAR15 (Karatzas, Gomez-Bigorda et al. 2015) : ICDAR15 ressemble beaucoup à ICDAR13 mais contient des images considérées plus difficiles. On y retrouve par exemple des images contenant du texte vu de côté ou des images où le texte est en diagonale. Cette base de données contient 4468 images d'entraînement et 2077 images pour le test.

5.4. Modèle

Dans ce chapitre, j'ai utilisé une version modifiée du modèle SATRN (Lee et al. 2020) qui était état de l'art sur la base de données SVT au moment du début de mes travaux. SATRN est un modèle image-vers-texte constitué d'une série de convolutions, d'un encodeur (image transformer) et d'un décodeur (text transformer). Un schéma de l'architecture de SATRN est présenté en figure 31.

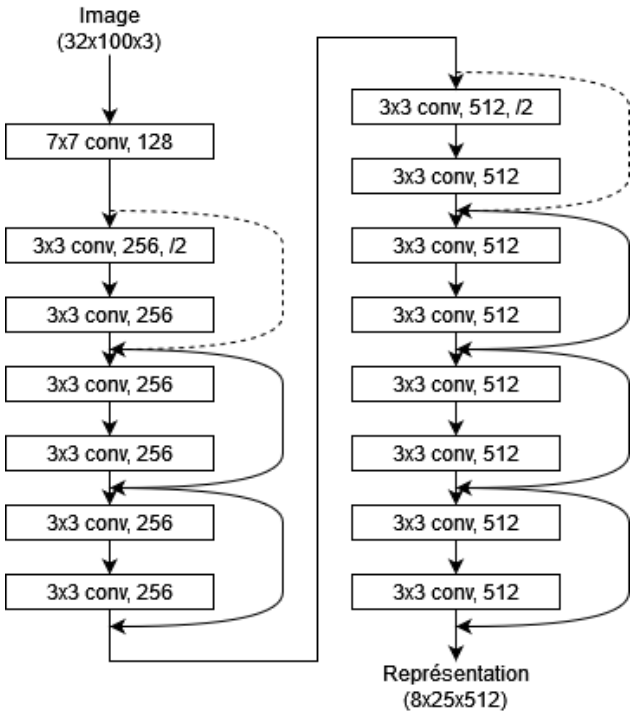


Figure 32 – Schéma de l'architecture des convolutions du modèle utilisé. Cette architecture est en grande partie inspirée de ResNet (K. He et al. 2016).

Mon modèle suit la même architecture générale que le modèle SATRN. La différence majeure concerne la série de convolutions qui est appliquée aux images avant le passage dans l'encodeur. Pour mon modèle, cette partie est plus complexe que dans l'article original afin de donner plus de capacité au modèle et ainsi lui permettre de condenser plus efficacement l'information contenue dans les images puisque dans mon cas, j'utilise des images plus variées. La figure 32 montre l'architecture des convolutions utilisée pour ma version du modèle. En dehors de cette modification sur les premières couches de convolutions, j'ai repris l'intégralité du modèle proposé par Lee et al. (2020).

5.5. Détails d'implémentation

5.5.1. Base de données réduite

Comme la quantité de données présente dans les différentes bases de données utilisées est très grande, j'ai décidé dans un premier temps de n'utiliser qu'une portion réduite des données. L'idée est de comparer différentes méthodes sur un dataset réduit afin de pouvoir faire tourner les expériences plus rapidement. Ainsi, je ne conserve qu'environ 2 millions

Base de données	Nombre d'exemples	Facteur multiplicatif
IAM	115 320*	×20
DocBank	2 000 000	-
SynthText	1 000 000	-
MJSynth	1 000 000	-

Tableau 16 – Description de la composition de la base de données réduite utilisée pour les expériences suivantes. "Nombre d'exemples" décrit le nombre d'exemple d'entraînement sélectionné dans chaque base de données et "Facteur multiplicatif" décrit le nombre de fois que les exemples issus d'une base de données sont répétés dans les données finales. * marque les datasets pour lesquels toutes les données disponibles sont utilisées.

d'exemples par catégorie de datasets (parmi texte manuscrit, texte tapuscript de type PDF et "Text in the Wild"). Comme certaines bases de données sont trop petites (ex : IAM), ces dernières sont dupliquées plusieurs fois dans les données d'entraînement afin d'équilibrer la quantité de données de chaque type. Le tableau 16 présente un récapitulatif de la quantité de données sélectionnée dans les différentes bases de données utilisées pour l'entraînement.

5.5.2. Implementation & hyperparamètres

Autant pendant l'entraînement que pendant la phase de test, le modèle et l'évaluation ne tiennent pas compte de la casse des mots. La raison principale est que certaines des bases de données utilisées n'en tiennent pas compte non plus et donc tout modèle entraîné sur ces datasets aura du mal à différencier entre majuscules et minuscules.

Les images sont redimensionnées de manière à ce qu'elles aient une hauteur uniforme tout en conservant le ratio largueur/hauteur initial. Pendant l'entraînement, une augmentation aléatoire est appliquée aux images. Cette augmentation est constituée d'une rotation aléatoire entre -10 et 10 degrés.

Le modèle est implémenté avec Pytorch. Il est entraîné sur 10 époques en conservant le meilleur modèle pour évaluation et en utilisant un optimisateur Adam (Kingma et Ba 2015). Les hyperparamètres les plus importants sont listés dans le tableau 17. Les hyperparamètres sont choisis de manière à maximiser les performances du modèle sur le corpus de développement. L'entraînement est réalisé avec 4× Nvidia Tesla T4 en utilisant la plateforme Grid5000 (Balouek et al. 2013).

Hyperparamètre	**Valeur**	**Hyperparamètre**	**Valeur**
Vitesse d'apprentissage	5e-6	Nb de couches transformer pour l'encodeur	12
Taille des mini-batches	16	Nb de couches transformer pour le décodeur	6
Hauteur des images	32	Nb de têtes d'attention pour les transformers	8

Tableau 17 – Liste des hyperparamètres pour mon implémentation du modèle SATRN.

5.5.3. Métriques

Les modèles sont évalués grâce à deux métriques : le CER (Character Error Rate) et l'"Accuracy". Dans le cadre de la reconnaissance optique de caractères, l'accuracy est la proportion de mots qui sont exactement reconnus par le modèle. Ainsi si N est la taille de l'ensemble sur lequel on évalue le modèle, y_i est le mot cible pour une image d'entrée x_i et $\hat{y}_i$ est le mot généré par le modèle pour cette même image x_i, alors :

$$accuracy = \frac{1}{N} \sum_{i=1}^{N} \delta_{y_i \hat{y}_i}$$

où $\delta_{y_i \hat{y}_i}$ est le symbole de Kronecker :

$$\delta_{y_i \hat{y}_i} = \begin{cases} 1 & \text{si } y_i = \hat{y}_i \\ 0 & \text{sinon} \end{cases}$$

Le problème avec la métrique accuracy, est qu'elle ne tient pas compte de la proximité du mot généré par rapport à la cible. Deux modèles qui génèrent respectivement les mots "tree" et "bard" là où le mot cible est "bird" auront donc le même score d'accuracy alors qu'intuitivement, le modèle qui s'est trompé d'un caractère seulement devrait être meilleur. Ainsi, de nombreux articles scientifiques sur le domaine de l'OCR utilisent également la métrique CER qui se calcule comme suit :

$$CER = \frac{S + D + I}{L}$$

Où L est la longueur du mot cible et S, D et I sont respectivement le nombre de substitutions, suppressions et insertions necessaires pour transformer le mot généré par le modèle

Datasets d'entraînement ↓ - Datasets d'évaluation →	IAM	Docbank	FUNSD
IAM	**80.1 (78.9)**	35.5 (30.4)	10.1 (9.5)
Docbank	1.8 (1.7)	**98.4 (98.3)**	35.1 (34.2)
MJSynth + SynthText	34.0 (33.2)	68.2 (61.7)	**41.7 (41.9)**
Tous (IAM + Docbank + MJSynth + SynthText)	**87.3 (86.0)**	**98.7** (95.0)	**49.3 (57.8)**

Datasets d'entraînement ↓ - Datasets d'évaluation →	SVT	IIIT5K	ICDAR13	ICDAR15
IAM	1.2 (1.2)	6.7 (7.9)	4.2 (4.2)	0.4 (0.8)
Docbank	2.7 (2.7)	11.8 (14.2)	8.9 (11.9)	2.5 (2.6)
MJSynth + SynthText	**83.1 (83.1)**	**86.1 (90.3)**	**82.5 (85.0)**	**68.6 (66.8)**
Tous (IAM + Docbank + MJSynth + SynthText)	**85.4 (85.4)**	**86.9 (91.6)**	**82.9 (86.3)**	**69.3 (69.0)**

Tableau 18 – Ce tableau montre les résultats d'accuracy (une accuracy plus élevée indique un meilleur modèle) obtenus par d'une part des modèles entraînés sur un seul type d'images (manuscrit, tapuscript ou *"text in the wild"*) et d'autre part un modèle unifié entraîné sur tous les types de données en même temps. Les scores sont calculés sur les ensembles de test des différentes bases de données. Les scores entre parenthèses indiquent les scores obtenus sur l'ensemble de validation.

vers le mot cible. La métrique CER peut également être vue comme la distance d'édition entre le mot généré par le modèle et le mot cible divisée par la longueur du mot cible.

D'autres travaux utilisent également la métrique WER (Word Error Rate) qui se calcule comme la métrique CER mais à l'échelle des mots (distance d'édition entre les mots d'une phrase générée par un modèle et d'une phrase cible divisé par le nombre de mots dans la phrase cible). Cependant, je n'utilise pas cette métrique dans les expériences qui vont suivre car je travaille exclusivement sur des images contenant un seul mot et la métrique WER est donc équivalente à l'accuracy.

Pour mes expériences, je ne prends pas en compte la casse du texte et mes modèles ainsi que mes métriques ne font pas la distinction entre des caractères minuscules ou majuscules.

5.6. Résultats

5.6.1. Premiers Résultats

Les tableaux 18 et 19 présentent les résultats obtenus par mes modèles sur les différentes bases de données de test (en termes d'*accuracy* et de CER respectivement). Je compare d'un côté les modèles entraînés avec un seul type de données et de l'autre un modèle entraîné avec

Datasets d'entraînement ↓ - Datasets d'évaluation →	IAM	Docbank	FUNSD
IAM	**8.4 (9.4)**	47.1 (54.1)	63.3 (64.0)
Docbank	85.6 (84.3)	**0.7 (0.6)**	36.7 (35.5)
MJSynth + SynthText	38.1 (39.7)	20.3 (26.5)	**32.7 (32.8)**
Tous (IAM + Docbank + MJSynth + SynthText)	**6.1 (7.1)**	0.5 (1.6)	**25.3 (21.7)**

Datasets d'entraînement ↓ - Datasets d'évaluation →	SVT	IIIT5K	ICDAR13	ICDAR15
IAM	81.6 (81.6)	76.9 (75.9)	79.1 (76.7)	84.5 (85.3)
Docbank	79.1 (79.1)	68.8 (64.4)	71.2 (64.1)	81.5 (81.1)
MJSynth + SynthText	**4.5 (4.5)**	**4.8 (3.1)**	**9.7 (6.4)**	**13.3 (13.9)**
Tous (IAM + Docbank + MJSynth + SynthText)	**3.7 (3.7)**	**4.4 (2.9)**	**8.7 (5.7)**	**13.5 (13.2)**

Tableau 19 – Ce tableau montre les résultats en termes de CER (un score plus faible indique un meilleur modèle) obtenus par d'une part des modèles entraînés sur un seul type d'images (manuscrit, tapuscript ou *"text in the wild"*) et d'autre part un modèle unifié entraîné sur tous les types de données en même temps. Les scores sont calculés sur les ensembles de test des différentes bases de données. Les scores entre parenthèses indiquent les scores obtenus sur l'ensemble de validation.

tous les types de données en même temps. On observe que le modèle entraîné avec tous les styles de données en même temps obtient une amélioration de performances sur toutes les bases de données de test. Les gains les plus significatifs sont obtenus sur IAM et FUNSD. Je montre donc ici que l'utilisation d'une plus grande diversité de données est bénéfique pour certaines bases de données de test.

5.6.2. Le problème des caractères spéciaux

Le tableau 20 montre les résultats obtenus par mon modèle unifié selon quels caractères sont considérés pendant l'évaluation. L'idée ici est de déterminer quels sont les types de caractères qui posent le plus de problèmes aux modèles dans le but de mieux comprendre pourquoi ils obtiennent des résultats très faibles sur FUNSD. J'effectue donc trois évaluations distinctes : une en calculant l'accuracy de manière normale, une où je ne prends en compte que les lettres et les chiffres (excluant ainsi tous les caractères spéciaux tels que les tirets et les ponctuations) et enfin une évaluation en ne tenant compte que des lettres. Sur la base de données FUNSD, les résultats sont plutôt flagrants. On remarque donc un gain de 20% d'accuracy (sur l'ensemble de test) lorsqu'on ne prend pas en compte les caractères spéciaux lors de l'évaluation montrant ainsi que mon modèle a beaucoup de difficultés à

Datasets d'évaluation ↓ - Métrique →	Tous	Lettres + chiffres	Lettres seulement
IAM	87.3 (86.0)	89.4 (88.2)	89.5 (88.4)
Docbank	98.7 (95.0)	99.1 (96.3)	99.2 (96.6)
FUNSD	49.3 (57.8)	73.3 (74.6)	78.9 (80.9)
SVT	85.4 (85.4)	85.4 (85.4)	85.4 (85.4)
IIIT5K	86.9 (91.6)	87.6 (92.3)	89.2 (92.6)
ICDAR13	82.9 (86.3)	83.9 (88.2)	85.7 (90.0)
ICDAR15	69.3 (69.0)	70.6 (70.1)	71.3 (72.2)

Tableau 20 – Ce tableau illustre les problèmes associés au manque de caractères spéciaux dans les données d'entraînement de notre modèle unifié. Trois variations de la métrique *accuracy* sont présentées selon que l'on considère tous les caractères, seulement les caractères alphanumériques ou seulement les lettres. Les scores sont calculés sur les ensembles de test des différentes bases de données. Les scores entre parenthèses indiquent les scores obtenus sur l'ensemble de validation.

les reconnaître. De plus, on gagne 5% en plus lorsqu'on de considère pas les chiffres. Cet effet des caractères spéciaux est plus ou moins marqué selon les bases de données et touche particulièrement FUNSD qui contient énormément de ponctuations notamment des deux-points puisque les documents contenus dans FUNSD sont des formulaires. A l'inverse, l'effet des caractères spéciaux est inexistant pour la base de données SVT qui ne contient aucun chiffre ou ponctuation. En analysant les données utilisées pour l'entraînement (càd. les trois datasets MJSynth, SynthText, Docbank et IAM), on remarque que ceux-ci contiennent peu ou pas du tout de caractères spéciaux. MJSynth et SynthText n'en contiennent pas du tout alors que Docbank et IAM contiennent quelques caractères spéciaux sous la forme de ponctuations mais avec une diversité assez faible ; la plupart étant des points ou des virgules.

Si on analyse un peu plus finement les cas où notre modèle fait des erreurs sur FUNSD, on remarque plusieurs cas de figure plutôt fréquents :

— Le cas d'erreur le plus courant est la présence non détectée d'une ponctuation à la fin d'un mot. Un exemple de ce genre d'erreur est illustré en figure 33(a). En effet, bien que ce ne soit pas le cas dans les bases de données utilisées pour l'entraînement, dans FUNSD, il est fréquent qu'un exemple soit constitué d'un mot suivi d'une ponctuation. Dans ce genre de cas, mon modèle a tendance à ne pas détecter la ponctuation ce qui compte alors comme une erreur du point de vue de l'accuracy même si le reste du mot est correctement détecté et retranscrit.

(a) “TO” **(b)** “BILDLE” **(c)** “IH”

Figure 33 – Illustration des principaux cas d’erreurs dans FUNSD. Les indication sous les images correspondent aux prédictions du modèle.

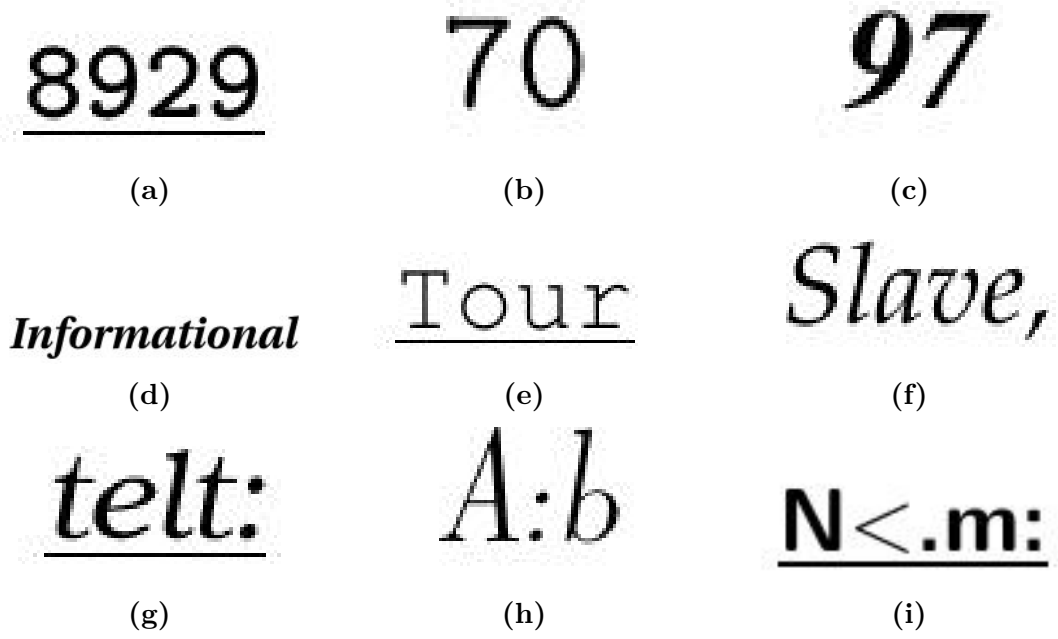
— Un deuxième cas courant montré en figure 33(b) concerne des séquences de chiffres (ou bien un mélange de chiffres et de lettres) qui sont reconnues par le modèle comme des lettres.

— Enfin, un grand nombre d’erreurs concernent aussi des caractères spéciaux qui n’apparaissent que très rarement dans les bases de données utilisées pour l’entraînement. C’est le cas notamment pour les chevrons (“<”, “>”). Dans ces cas, comme illustré en figure 33(c), le modèle prédit le plus souvent une suite de lettres puisqu’il n’a pas vu suffisamment d’exemples d’entraînement contenant ces caractères pour savoir les reconnaître.

En plus de ce biais de sous-représentation de certains caractères, il existe aussi un biais sur les polices de caractères utilisées. En effet, la base de données d’entraînement utilisée pour représenter les documents tapuscripts, Docbank, est extraite d’articles scientifiques issus de la base d’articles arxiv. Or, les polices de caractères utilisées pour la rédaction d’articles scientifiques sont assez uniformes dans l’ensemble, ce qui résulte en une base de données contenant principalement la même police dans tous les exemples d’entraînement. Ce manque de diversité des polices de caractères dans les données pose encore une fois problème lorsqu’on essaye par la suite d’appliquer ce modèle à des bases de données comme FUNSD qui contiennent des polices très différentes (une police à largeur fixe dans le cas de FUNSD).

Je me suis donc tourné vers des techniques d’augmentation de données pour combler les manques des bases de données d’entraînement précédemment utilisées. On commence par générer aléatoirement une liste de *mots* avec 4 méthodes différentes :

— Dans 10% des cas, on génère une suite aléatoire de caractères. Les caractères peuvent être des lettres (minuscules ou majuscules), des chiffres ou bien des caractères spéciaux. La liste des caractères spéciaux possibles est assez large et contient par exemple



(a) **(b)** **(c)**

(d) **(e)** **(f)**

(g) **(h)** **(i)**

Figure 34 – Mots générés par le système de d'augmentation des données tapuscrites

des chevrons, des parenthèses, des symboles arithmétiques, etc. Cette génération permet d'introduire dans le dataset d'entraînement des codes comme "84-C-7" qui est un exemple réel issu de la base de données FUNSD. Ceci permet donc à notre modèle d'avoir accès à plus d'exemples qui ne correspondent à aucun mot connu et qui mélangent lettres, chiffres et caractères spéciaux.

— Ensuite, dans 20% des cas, on génère une suite de 1 à 6 chiffres. Le nombre de chiffres est choisi au hasard de manière équiprobable. Le but est d'amener plus d'exemples d'entraînement contenant des chiffres car ceux-ci sont assez peu présents dans les bases de données utilisées pour l'entraînement. Ils sont aussi une source importante d'erreurs sur FUNSD comme indiqué par la figure 20.

— Finalement, j'utilise le dictionnaire anglais de la bibliothèque python NLTK (Bird, E. Klein et Loper 2009) pour y sélectionner des mots aléatoirement. Ces mots sont dans des proportions égales soit tout en minuscules, tout en majuscules ou avec seulement la première lettre en majuscule (couvrant ainsi la majorité des cas d'usage). De plus, dans 20% des cas, on fait suivre le mot d'une ponctuation aléatoire parmi : ".", ",", " :", " !" et " ?". Encore une fois, cet ajout de ponctuations permet de proposer des exemples correspondant à certains des cas d'erreurs les plus courants dans FUNSD.

Une fois la liste de mots générée, j'utilise LaTeX [2] pour produire une image contenant uniquement ce mot. La police utilisée est choisie aléatoirement parmi les options par défault proposées par LaTeX. Cet ensemble de polices, bien que restreint, est plutôt varié (polices avec espacement fixe ou non, avec ou sans serifs, etc.). On choisit également aléatoirement si on affiche le mot en gras, en italique ou en souligné.

Je génère avec ce processus 100 000 exemples d'entraînement. Finalement, comme dans une situation réelle, les documents considérés sont souvent des documents scannés ou issus de photos, ceux-ci sont souvent très bruités. On duplique donc notre base de données en y ajoutant une version bruitée de chaque exemple. Pour le moment, nos images générées ne contiennent que des pixels complètement blancs ou complètement noirs ce qui n'est jamais le cas quand on a affaire à un pdf réel issu d'un scan ou d'une photo. On commence donc par changer la couleur des pixels blancs et noirs de l'image pour leur donner une couleur plus naturelle. Pour ce faire, on commence par choisir 2 valeurs aléatoires définissant des nouvelles valeurs minimales et maximales pour les pixels de l'image.

$$b = randint(0, 55)$$

$$w = randint(245, 255)$$

où $randint(i, j)$ correspond à une valeur entière aléatoire entre i et j. Pour une valeur initiale x d'un pixel de l'image, la nouvelle valeur x' est alors :

$$x' = \frac{w - b}{255} \times x + b$$

On ajoute ensuite dans 50% des cas un bruit gaussien avec une moyenne nulle et une variance aléatoire (entre 0 et 0.005) et/ou dans 50% des cas un bruit *salt and pepper*. A la suite de cette augmentation, on obtient donc 200 000 images dont 100 000 sont bruitées. Ce dataset généré est beaucoup plus varié en termes de forme du texte (police, gras, souligné, etc.) et de fond (plus de caractères spéciaux, séquences de lettres aléatoires, etc.) par rapport aux bases de données que j'utilisais jusqu'ici.

J'entraîne une nouvelle fois le modèle mais cette fois en remplaçant la moitié des données de Docbank par nos nouvelles données synthétiques (dupliquées 5 fois). Ceci permet de maintenir constante la quantité globale de données utilisées pour l'entraînement tout en

2. `https://www.latex-project.org/`

Datasets d'évaluation ↓ - Modèle →	Baseline	Baseline + T
IAM	**87.3** (**86.0**)	86.7 (85.6)
Docbank	**98.7** (**95.0**)	97.8 (93.8)
FUNSD	49.3 (57.8)	**63.4** (**67.0**)
SVT	85.4 (85.4)	**90.1** (**88.1**)
IIIT5K	**86.9** (**91.6**)	86.0 (91.0)
ICDAR13	82.9 (86.3)	**84.3** (**88.0**)
ICDAR15	69.3 (**69.0**)	**70.4** (**69.1**)

Tableau 21 – Ce tableau montre les résultats en termes d'*accuracy* d'un modèle entraîné avec une portion de données tapuscriptes synthétiques ("Baseline + T") comparé à notre baseline (SATRN entraîné sur IAM, Docbank, SynthText et MJSynth).

diversifiant les données tapuscriptes. Le tableau 21 présente les résultats obtenus. On remarque une amélioration des performances sur certaines bases de données et une diminution sur d'autres. Cependant, pour les bases de données sur lesquelles j'enregistre une diminution des performances, cette diminution se limite à moins de 1%. A l'inverse, on observe un gain de performances beaucoup plus significatif sur SVT et FUNSD. SVT profite probablement plus de la diversification des polices (puisque le dataset ne contient pas de caractères spéciaux). Ces résultats montrent l'intérêt d'une plus grande diversité dans les données utilisées pour l'entraînement.

5.7. Conclusion

Dans ce chapitre, j'ai étudié l'effet d'un entraînement des modèles d'OCR sur un ensemble d'entraînement varié, constitué d'un éventail de sous-tâches allant de bases de données d'écriture manuscrite comme IAM, à des bases de données de type "*text in the wild*". J'ai donc entraîné un modèle multitâche en m'inspirant du modèle SATRN (Lee et al. 2020), qui était état de l'art au moment de commencer mes expériences.

Mes expériences ont montré qu'il y a en effet un intérêt certain à entraîner un modèle avec des données plus variées, mon modèle multitâche ayant des performances accrues sur la plupart des bases de données de test étudiées comparé à des modèles entraînés sur un seul type de données. J'ai de plus mis en évidence que les bases de données que j'utilise ne contiennent que très peu de caractères spéciaux et j'ai montré qu'il est possible d'améliorer encore les performances des modèles sur certaines bases de données comme FUNSD

(qui contient beaucoup de caractères spéciaux) en générant automatiquement des données synthétiques contenant plus de tels caractères.

Cette partie de ma thèse, pourrait faire l'objet de beaucoup de futurs travaux. Je n'ai par exemple travaillé que sur des bases de données réduites, et il serait intéressant de voir si les observations que j'ai émises se généralisent lorsqu'on utilise les bases de données complètes. De plus, depuis que j'ai commencé à travailler sur SATRN, de nombreux modèles plus performants ont vu le jour et la méthode présentée pourrait également être appliquée à ces nouveaux modèles.

Conclusion

Dans cette thèse, j'ai présenté plusieurs méthodes et architectures permettant de traiter les biais qui peuvent être présents dans les bases de données de TAL. C'est ce problème des biais dans les données qui a guidé mon projet de recherche. Le point d'origine de cette réflexion, est la conclusion du chapitre 2 dans lequel, après plusieurs expériences infructueuses, j'ai mis en lumière que les bases de données de question-réponse que j'utilisais à l'époque contiennent des biais importants probablement issus du processus d'annotation des données. Ces biais, qu'il est souvent difficile de séparer de l'information pertinente, pénalisent l'entraînement de modèles d'apprentissage profond. De plus, l'élimination de ces biais par un processus automatique n'est pas aisée tant la limite entre ce qui est un biais et ce qui est une information pertinente est floue et parfois même dépendante de la tâche considérée.

Premièrement, dans le chapitre 3, je présente des travaux réalisés sur la tâche de résumé automatique. Sur cette tâche sont présents différents biais dépendant de la manière d'écrire de chacun des annotateurs. Sur Gigaword (Rush, Chopra et Weston 2015), on trouve donc par exemple des résumés plus ou moins longs et détaillés ou encore une variété de résumés écrits à la voix active ou passive. Tous ces résumés sont valides, mais leur diversité ajoute de la confusion et rend l'apprentissage des modèles plus difficile. Je propose une architecture permettant de capturer cette diversité dans les résumés. Cette architecture et la méthode d'apprentissage associée permettent à un modèle de générer plusieurs résumés pour une même phrase d'entrée à l'aide de plusieurs décodeurs distincts. Ces différents décodeurs se spécialisent chacun dans un style d'écriture tout en essayant à eux tous de couvrir au maximum l'espace des possibles, à la manière de l'algorithme espérance-maximisation. Il est dès lors possible pour un utilisateur de contrôler la génération en choisissant le style de résumé qu'il préfère. Ce travail se retrouve au croisement de plusieurs problématiques importantes pour le résumé automatique. En plus du problème des biais, je traite également

de la manière de capturer la variabilité dans les résumés en proposant une alternative aux méthodes génératives utilisées précédemment. De plus, mon travail s'inscrit dans une longue liste de travaux proposant des approches pour rendre les résumés contrôlables (Guo, Y. Du et Zhao 2021 ; Fan, Grangier et Auli 2018).

Ensuite, dans le chapitre 4, je me suis intéressé à la tâche de question-réponse à choix multiples et j'ai présenté une méthode non supervisée de génération automatique de questions. Cette méthode permet de limiter certains biais présents habituellement dans les données comme ceux issus des annotateurs. J'ai dans un premier temps appliqué cette méthode à l'entraînement non supervisé d'un système de question-réponse. Cependant, c'est en utilisant ce type de données pour l'affinage de modèles déjà existants qu'on obtient les résultats les plus intéressants. En effet, j'ai montré dans ce chapitre qu'il est possible d'utiliser des questions générées par un processus de génération simple pour affiner un modèle multitâche – UnifiedQA (Khashabi, Min et al. 2020) – sur de nouveaux domaines et bases de données. Ma méthode permet d'obtenir une augmentation de performances significative comparée à un modèle UnifiedQA qui n'a pas été affiné. Ce genre de méthodes qui visent à utiliser de plus en plus de données issues de différentes sources pour créer des modèles généraux est très en vogue actuellement dans de nombreux domaines. Néanmoins, l'intérêt de tels modèles diminue s'il est nécessaire d'affiner le modèle avec des données annotées sur chaque nouvelle tâche pour obtenir les meilleures performances. Mon approche de génération de questions permet alors d'affiner ces modèles sans avoir recours à des données annotées coûteuses à obtenir.

Finalement, inspiré par UnifiedQA, qui est un modèle multitâche entraîné sur un grand nombre de tâches de question-réponse, je me suis attelé à la création d'un modèle similaire pour la tâche de reconnaissance optique de texte (OCR). Ce genre de modèle permet de diluer les biais spécifiques à chaque base de données dans l'ensemble des données d'entraînement. J'ai donc réuni plusieurs bases de données d'OCR parmi les plus utilisées (à la fois des données manuscrites et tapuscrites) et je les ai combinées pour entraîner un modèle inspiré de SATRN (Lee et al. 2020). Mes expériences ont montré une augmentation significative des performances lorsque j'entraîne un modèle unifié comparé à des modèles entraînés sur une tâche spécifique. De plus, j'ai montré que les bases de données utilisées ne couvrent pas tous les cas d'usage et notamment, qu'elles contiennent assez peu de caractères spéciaux

(points, virgules, parenthèses, etc.). J'ai démontré que de meilleures performances peuvent être obtenues en générant procéduralement des exemples d'entraînement contenant ce genre de caractères. Cette approche a amené des résultats prometteurs qui montrent qu'à l'instar de UnifiedQA pour le question-réponse, il est possible d'atteindre un meilleur niveau de généralisation en combinant ensemble plusieurs bases de données d'OCR.

Dans ces trois volets, j'explore donc différentes facettes du biais avec comme support diverses tâches de TAL. Chacune de mes approches a été réalisée en m'appuyant sur les méthodes état de l'art de l'époque. Cependant, le domaine du TAL évolue très vite et certaines des approches proposées mériteraient d'être réétudiées aujourd'hui, à la lumière des dernières avancées. Par exemple, dans mes travaux les plus anciens, réalisés sur le résumé automatique, j'utilise des modèles séquences-à-séquence construits à partir d'un LSTM. Nous pourrions alors nous demander quels seraient les résultats de cette approche si nous la transposions à des architectures plus contemporaines comme des transformers ou même des modèles pré-entraînés. De plus, en ce qui concerne le chapitre 4 (question-réponse) et le chapitre 5 (OCR), et même s'il s'agit de travaux plus récents, il reste de nombreuses pistes de recherche à explorer. Pour la partie question-réponse, il serait par exemple intéressant de tester l'approche proposée avec d'autres méthodes de génération de questions ou sur d'autres modèles état de l'art. Et du côté de la reconnaissance de texte, de nombreuses choses restent à être finalisées. Il me faudra donc dans le futur compléter cette partie en réalisant notamment les expériences sur les bases de données complètes (à la place des bases de données réduites que j'utilise actuellement) et en testant l'approche sur les modèles état de l'art actuels.

Pour conclure, je dirais qu'à la lumière des dernières avancées en intelligence artificielle avec spécifiquement l'arrivée de ChatGPT qui a fait beaucoup parler de lui dans les médias, le problème des biais me parait de plus en plus important à résoudre. Alors que jusque-là les modèles d'intelligence artificielle étaient limités à la recherche et à seulement quelques applications notables dans l'industrie, nous voyons ces dernières années de plus en plus de modèles mis à la disposition du public. Ces modèles sont ainsi utilisés pour diverses applications plus ou moins importantes. Il est alors de la responsabilité du monde scientifique de s'assurer que les décisions prises par ces modèles sont guidés par des faits rationnels et non par des biais présents initialement dans les données d'entraînement.

Liste des publications

---

(1) Guillaume Le Berre, Christophe Cerisara et al. (mai 2022). "Unsupervised multiple-choice question generation for out-of-domain Q&A fine-tuning". In : *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*. Dublin, Ireland : Association for Computational Linguistics, p. 732-738. DOI : 10.18653/v1/2022.acl-short.83. URL : https://aclanthology.org/2022.acl-short.83

(2) Christophe Cerisara, Paul Caillon et Guillaume Le Berre (2021). "Unsupervised Post-Tuning of Deep Neural Networks". In : *2021 International Joint Conference on Neural Networks (IJCNN)*, p. 1-8. DOI : 10.1109/IJCNN52387.2021.9534198

(3) Guillaume Le Berre et Philippe Langlais (2020). "Attending Knowledge Facts with BERT-like Models in Question-Answering : Disappointing Results and Some Explanations". In : *Canadian Conference on Artificial Intelligence*. Springer, p. 356-367

(4) Guillaume Le Berre et Christophe Cerisara (2020). "Seq-to-NSeq model for multi-summary generation". In : *ESANN 2020*

Bibliographie

Adeli, Ehsan et al. (2020). *Bias-Resilient Neural Network*. URL : https://openreview.net/forum?id=Bke8764twr.

Antol, Stanislaw et al. (2015). "VQA : Visual Question Answering". In : *International Conference on Computer Vision (ICCV)*.

Ayana et al. (2016). "Neural Headline Generation with Minimum Risk Training". In : *CoRR* abs/1604.01904. arXiv : 1604.01904. URL : http://arxiv.org/abs/1604.01904.

Bahdanau, Dzmitry, Kyung Hyun Cho et Yoshua Bengio (jan. 2015). "Neural machine translation by jointly learning to align and translate". English (US). In : 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date : 07-05-2015 Through 09-05-2015.

Bai, Yang et Daisy Zhe Wang (2021). "More Than Reading Comprehension : A Survey on Datasets and Metrics of Textual Question Answering". In : *CoRR* abs/2109.12264. arXiv : 2109.12264. URL : https://arxiv.org/abs/2109.12264.

Balouek, Daniel et al. (2013). "Adding Virtualization Capabilities to the Grid'5000 Testbed". In : *Cloud Computing and Services Science*. Sous la dir. d'Ivan I. Ivanov et al. T. 367. Communications in Computer and Information Science. Springer International Publishing, p. 3-20. ISBN : 978-3-319-04518-4. DOI : 10.1007/978-3-319-04519-1_1.

Banijamali, Ershad, Ali Ghodsi et Pascal Poupart (2017). "Generative mixture of networks". In : *2017 International Joint Conference on Neural Networks (IJCNN)*, p. 3753-3760.

Banko, Michele, Vibhu O. Mittal et Michael J. Witbrock (oct. 2000). "Headline Generation Based on Statistical Translation". In : *Proceedings of the 38th Annual Meeting*

of the Association for Computational Linguistics. Hong Kong : Association for Computational Linguistics, p. 318-325. DOI : 10.3115/1075218.1075259. URL : https://aclanthology.org/P00-1041.

BAUTISTA, Darwin et Rowel ATIENZA (2022). "Scene text recognition with permuted autoregressive sequence models". In : *Computer Vision–ECCV 2022 : 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*. Springer, p. 178-196.

BENGIO, Yoshua, Réjean DUCHARME et Pascal VINCENT (2000). "A Neural Probabilistic Language Model". In : *Advances in Neural Information Processing Systems*. Sous la dir. de T. LEEN, T. DIETTERICH et V. TRESP. T. 13. MIT Press. URL : https://proceedings.neurips.cc/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

BIRD, Steven, Ewan KLEIN et Edward LOPER (2009). *Natural language processing with Python : analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

BOWMAN, Samuel R. et al. (sept. 2015). "A large annotated corpus for learning natural language inference". In : *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal : Association for Computational Linguistics, p. 632-642. DOI : 10.18653/v1/D15-1075. URL : https://www.aclweb.org/anthology/D15-1075.

BROWN, Tom et al. (2020). "Language Models are Few-Shot Learners". In : *Advances in Neural Information Processing Systems*. Sous la dir. de H. LAROCHELLE et al. T. 33. Curran Associates, Inc., p. 1877-1901. URL : https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

CERISARA, Christophe, Paul CAILLON et Guillaume LE BERRE (2021). "Unsupervised Post-Tuning of Deep Neural Networks". In : *2021 International Joint Conference on Neural Networks (IJCNN)*, p. 1-8. DOI : 10.1109/IJCNN52387.2021.9534198.

CHAUDHARY, Kartik et Raghav BALI (2022). "Easter2. 0 : Improving convolutional models for handwritten text recognition". In : *arXiv preprint arXiv :2205.14879*.

CHOPRA, Sumit, Michael AULI et Alexander M. RUSH (2016). "Abstractive Sentence Summarization with Attentive Recurrent Neural Networks". In : *Proceedings of the 2016*

Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies. San Diego, California : Association for Computational Linguistics, p. 93-98. DOI : 10.18653/v1/N16-1012. URL : http://aclweb.org/anthology/N16-1012.

CLARK, Christopher et al. (juin 2019). "BoolQ : Exploring the Surprising Difficulty of Natural Yes/No Questions". 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)*. Minneapolis, Minnesota : Association for Computational Linguistics, p. 2924-2936. DOI : 10.18653/v1/N19-1300. URL : https://aclanthology.org/N19-1300.

CLARK, Peter, Isaac COWHEY et al. (mars 2018). "Think you have Solved Question Answering ? Try ARC, the AI2 Reasoning Challenge". In : URL : http://arxiv.org/abs/1803.05457.

CLARK, Peter, Oren ETZIONI et al. (déc. 2020). "From 'F' to 'A' on the N.Y. Regents Science Exams : An Overview of the Aristo Project". In : *AI Magazine* 41.4, p. 39-53. DOI : 10.1609/aimag.v41i4.5304. URL : https://ojs.aaai.org/index.php/aimagazine/article/view/5304.

COQUENET, Denis, Clément CHATELAIN et Thierry PAQUET (2022). "End-to-end handwritten paragraph text recognition using a vertical attention network". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1, p. 508-524.

CUI, Mengmeng et al. (2021). "Representation and Correlation Enhanced Encoder-Decoder Framework for Scene Text Recognition". In : *Document Analysis and Recognition–ICDAR 2021 : 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part IV*, p. 156-170.

DAS, Rubel et al. (nov. 2016). "A rule based question generation framework to deal with simple and complex sentences". In : Institute of Electrical et Electronics Engineers Inc., p. 542-548. ISBN : 9781509020287. DOI : 10.1109/ICACCI.2016.7732102.

DASIGI, Pradeep et al. (nov. 2019). "Quoref : A Reading Comprehension Dataset with Questions Requiring Coreferential Reasoning". 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)*. Hong Kong, China :

Association for Computational Linguistics, p. 5925-5932. DOI : 10.18653/v1/D19-1606. URL : https://aclanthology.org/D19-1606.

Devlin, Jacob et al. (juin 2019). "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)*. Minneapolis, Minnesota : Association for Computational Linguistics, p. 4171-4186. DOI : 10.18653/v1/N19-1423. URL : https://aclanthology.org/N19-1423.

Dorr, Bonnie, David Zajic et Richard Schwartz (2003). "Hedge Trimmer : A Parse-and-Trim Approach to Headline Generation". In : *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, p. 1-8. URL : https://aclanthology.org/W03-0501.

Du, Xinya, Junru Shao et Claire Cardie (juill. 2017). "Learning to Ask : Neural Question Generation for Reading Comprehension". In : *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. Vancouver, Canada : Association for Computational Linguistics, p. 1342-1352. DOI : 10.18653/v1/P17-1123. URL : https://aclanthology.org/P17-1123.

Dua, Dheeru et al. (juin 2019). "DROP : A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs". 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)*. Minneapolis, Minnesota : Association for Computational Linguistics, p. 2368-2378. DOI : 10.18653/v1/N19-1246. URL : https://aclanthology.org/N19-1246.

Fabbri, Alexander et al. (juill. 2020). "Template-Based Question Generation from Retrieved Sentences for Improved Unsupervised Question Answering". In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online : Association for Computational Linguistics, p. 4508-4513. DOI : 10.18653/v1/2020.acl-main.413. URL : https://aclanthology.org/2020.acl-main.413.

Fan, Angela, David Grangier et Michael Auli (juill. 2018). "Controllable Abstractive Summarization". In : *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Melbourne, Australia : Association for Computational Linguistics, p. 45-54. DOI : 10.18653/v1/W18-2706. URL : https://aclanthology.org/W18-2706.

Freitag, Markus et Scott Roy (avr. 2018). "Unsupervised Natural Language Generation with Denoising Autoencoders". In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, p. 3922-3929. URL : http://arxiv.org/abs/1804.07899.

Gehring, Jonas et al. (juin 2017). "Convolutional Sequence to Sequence Learning". In : *Proceedings of the 34th International Conference on Machine Learning*. Sous la dir. de Doina Precup et Yee Whye Teh. T. 70. Proceedings of Machine Learning Research. PMLR, p. 1243-1252. URL : https://proceedings.mlr.press/v70/gehring17a.html.

Geva, Mor, Yoav Goldberg et Jonathan Berant (nov. 2019). "Are We Modeling the Task or the Annotator ? An Investigation of Annotator Bias in Natural Language Understanding Datasets". 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)*. Hong Kong, China : Association for Computational Linguistics, p. 1161-1166. DOI : 10.18653/v1/D19-1107. URL : https://aclanthology.org/D19-1107.

Ghaddar, Abbas et al. (déc. 2022). "Revisiting Pre-trained Language Models and their Evaluation for Arabic Natural Language Processing". In : *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates : Association for Computational Linguistics, p. 3135-3151. URL : https://aclanthology.org/2022.emnlp-main.205.

Goodfellow, Ian, Yoshua Bengio et Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Goodfellow, Ian, Jean Pouget-Abadie et al. (2014). "Generative Adversarial Nets". In : *Advances in Neural Information Processing Systems*. Sous la dir. de Z. Ghahramani et al. T. 27. Curran Associates, Inc. URL : https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Guo, Xiaojie, Yuanqi Du et Liang Zhao (2021). "Property Controllable Variational Autoencoder via Invertible Mutual Dependence". In : *ICLR*. URL : https://openreview.net/forum?id=tYxG_OMs9WE.

Gupta, Ankush, Andrea Vedaldi et Andrew Zisserman (2016). “Synthetic Data for Text Localisation in Natural Images”. In : *IEEE Conference on Computer Vision and Pattern Recognition*.

Gururangan, Suchin et al. (juin 2018). “Annotation Artifacts in Natural Language Inference Data”. In : *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana : Association for Computational Linguistics, p. 107-112. doi : 10.18653/v1/N18-2017. url : https://aclanthology.org/N18-2017.

He, Kaiming et al. (2016). “Deep Residual Learning for Image Recognition”. In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770-778.

He, Yue et al. (2022). “Visual Semantics Allow for Textual Reasoning Better in Scene Text Recognition”. In : 36.

Hochreiter, Sepp et Jürgen Schmidhuber (1997a). “Long Short-Term Memory”. In.

— (nov. 1997b). “Long Short-Term Memory”. In : *Neural Computation* 9.8, p. 1735-1780. issn : 0899-7667. doi : 10.1162/neco.1997.9.8.1735. eprint : https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. url : https://doi.org/10.1162/neco.1997.9.8.1735.

Huang, Zixian et al. (juill. 2022). “Clues Before Answers : Generation-Enhanced Multiple-Choice QA”. In : *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. Seattle, United States : Association for Computational Linguistics, p. 3272-3287. doi : 10.18653/v1/2022.naacl-main.239. url : https://aclanthology.org/2022.naacl-main.239.

Hudson, Drew Arad et Christopher D. Manning (2018). “Compositional Attention Networks for Machine Reasoning”. In : *International Conference on Learning Representations*. url : https://openreview.net/forum?id=S1Euwz-Rb.

Jaderberg, Max et al. (2014). “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition”. In : *Workshop on Deep Learning, NIPS*.

Jaume, Guillaume, Hazim Kemal Ekenel et Jean-Philippe Thiran (2019). “FUNSD : A Dataset for Form Understanding in Noisy Scanned Documents”. In : *2019 International*

Conference On Document Analysis And Recognition Workshops (Icdarw) And 2Nd International Workshop On Open Services And Tools For Document Analysis (Ost), Vol 2. CONF. IEEE, p. 1-6.

Johnson, Justin et al. (2017). "Clevr : A diagnostic dataset for compositional language and elementary visual reasoning". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2901-2910.

Kamal Eddine, Moussa, Antoine Tixier et Michalis Vazirgiannis (nov. 2021). "BARThez : a Skilled Pretrained French Sequence-to-Sequence Model". In : *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online et Punta Cana, Dominican Republic : Association for Computational Linguistics, p. 9369-9390. doi : 10.18653/v1/2021.emnlp-main.740. url : https://aclanthology.org/2021.emnlp-main.740.

Karatzas, Dimosthenis, Lluis Gomez-Bigorda et al. (2015). "ICDAR 2015 competition on Robust Reading". In : *13th International Conference on Document Analysis and Recognition, ICDAR 2015, Nancy, France, August 23-26, 2015*. IEEE Computer Society, p. 1156-1160. doi : 10.1109/ICDAR.2015.7333942. url : https://doi.org/10.1109/ICDAR.2015.7333942.

Karatzas, Dimosthenis, Faisal Shafait et al. (2013). "ICDAR 2013 Robust Reading Competition". In : *2013 12th International Conference on Document Analysis and Recognition*, p. 1484-1493. doi : 10.1109/ICDAR.2013.221.

Khashabi, Daniel, Yeganeh Kordi et Hannaneh Hajishirzi (2022). "Unifiedqa-v2 : Stronger generalization via broader cross-format training". In : *arXiv preprint arXiv :2202.12359*.

Khashabi, Daniel, Sewon Min et al. (nov. 2020). "UNIFIEDQA : Crossing Format Boundaries with a Single QA System". In : *Findings of the Association for Computational Linguistics : EMNLP 2020*. Online : Association for Computational Linguistics, p. 1896-1907. doi : 10.18653/v1/2020.findings-emnlp.171. url : https://aclanthology.org/2020.findings-emnlp.171.

Khot, Tushar et al. (2020). "Qasc : A dataset for question answering via sentence composition". In : *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 34. 05, p. 8082-8090.

Kim, Byungju et al. (2019). “Learning not to learn : Training deep neural networks with biased data”. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 9012-9020.

Kingma, Diederik P. et Jimmy Ba (2015). “Adam : A Method for Stochastic Optimization”. In : *ICLR (Poster)*. URL : http://arxiv.org/abs/1412.6980.

Kingma, Diederik P. et Max Welling (2014). “Auto-Encoding Variational Bayes”. In : *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv : http://arxiv.org/abs/1312.6114v10 [stat.ML].

Kitaev, Nikita, Lukasz Kaiser et Anselm Levskaya (2020). “Reformer : The Efficient Transformer”. In : *International Conference on Learning Representations*. URL : https://openreview.net/forum?id=rkgNKkHtvB.

Klein, Guillaume et al. (juill. 2017). “OpenNMT : Open-Source Toolkit for Neural Machine Translation”. In : *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada : Association for Computational Linguistics, p. 67-72. URL : https://aclanthology.org/P17-4012.

Kočiský, Tomáš et al. (2018). “The NarrativeQA Reading Comprehension Challenge”. In : *Transactions of the Association for Computational Linguistics* 6, p. 317-328. DOI : 10.1162/tacl_a_00023. URL : https://aclanthology.org/Q18-1023.

Kwiatkowski, Tom et al. (2019). “Natural Questions : A Benchmark for Question Answering Research”. In : *Transactions of the Association for Computational Linguistics* 7, p. 452-466. DOI : 10.1162/tacl_a_00276. URL : https://aclanthology.org/Q19-1026.

Lai, Guokun et al. (sept. 2017). “RACE : Large-scale ReAding Comprehension Dataset From Examinations”. In : *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark : Association for Computational Linguistics, p. 785-794. DOI : 10.18653/v1/D17-1082. URL : https://aclanthology.org/D17-1082.

Lan, Zhenzhong et al. (2020). “ALBERT : A Lite BERT for Self-supervised Learning of Language Representations”. In : *International Conference on Learning Representations*. URL : https://openreview.net/forum?id=H1eA7AEtvS.

Lapalme, Guy (nov. 2019). “Realizing Universal Dependencies Structures using a symbolic approach”. In : *The Second Multilingual Surface Realisation Shared Task (SR’19) :*

Overview and Evaluation Results. In Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR), (EMNLP-2019). Sous la dir. de Simon Mille et al. ACL. Hong-Kong, 8 pages.

Lapalme, Guy (jan. 2021). "The jsRealB Text Realizer : Organization and Use Cases". In : arXiv :2012.15425. URL : https://arxiv.org/abs/2012.15425.

Le, Hang et al. (mai 2020). "FlauBERT : Unsupervised Language Model Pre-training for French". English. In : *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France : European Language Resources Association, p. 2479-2490. ISBN : 979-10-95546-34-4. URL : https://aclanthology.org/2020.lrec-1.302.

Le Berre, Guillaume et Christophe Cerisara (2020). "Seq-to-NSeq model for multi-summary generation". In : *ESANN 2020*.

Le Berre, Guillaume, Christophe Cerisara et al. (mai 2022). "Unsupervised multiple-choice question generation for out-of-domain Q&A fine-tuning". In : *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*. Dublin, Ireland : Association for Computational Linguistics, p. 732-738. DOI : 10.18653/v1/2022.acl-short.83. URL : https://aclanthology.org/2022.acl-short.83.

Le Berre, Guillaume et Philippe Langlais (2020). "Attending Knowledge Facts with BERT-like Models in Question-Answering : Disappointing Results and Some Explanations". In : *Canadian Conference on Artificial Intelligence*. Springer, p. 356-367.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In : *Proceedings of the IEEE* 86.11, p. 2278-2324.

Lee, Junyeop et al. (2020). "On Recognizing Texts of Arbitrary Shapes with 2D Self-Attention". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Text and Documents in the Deep Learning Era WTDDLE*.

Levy, Omer et al. (mai 2015). "Do Supervised Distributional Methods Really Learn Lexical Inference Relations ?" In : *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. Denver, Colorado : Association for Computational Linguistics, p. 970-976. DOI : 10.3115/v1/N15-1098. URL : https://aclanthology.org/N15-1098.

Lewis, Patrick, Ludovic Denoyer et Sebastian Riedel (juill. 2019). “Unsupervised Question Answering by Cloze Translation”. In : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy : Association for Computational Linguistics, p. 4896-4910. doi : 10.18653/v1/P19-1484. url : https://aclanthology.org/P19-1484.

Li, Minghao et al. (déc. 2020). “DocBank : A Benchmark Dataset for Document Layout Analysis”. In : *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online) : International Committee on Computational Linguistics, p. 949-960. doi : 10.18653/v1/2020.coling-main.82. url : https://aclanthology.org/2020.coling-main.82.

Li, Zhongli et al. (juill. 2020). “Harvesting and Refining Question-Answer Pairs for Unsupervised QA”. In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online : Association for Computational Linguistics, p. 6719-6728. doi : 10.18653/v1/2020.acl-main.600. url : https://aclanthology.org/2020.acl-main.600.

Liang, Chen et al. (mai 2018). “Distractor Generation for Multiple Choice Questions Using Learning to Rank”. In : Association for Computational Linguistics (ACL), p. 284-290. doi : 10.18653/v1/w18-0533. url : https://www.aclweb.org/anthology/W18-0533.

Lin, Chin-Yew (juill. 2004). “ROUGE : A Package for Automatic Evaluation of Summaries”. In : *Text Summarization Branches Out*. Barcelona, Spain : Association for Computational Linguistics, p. 74-81. url : https://aclanthology.org/W04-1013.

Lin, Kevin et al. (nov. 2019). “Reasoning Over Paragraph Effects in Situations”. In : *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Hong Kong, China : Association for Computational Linguistics, p. 58-62. doi : 10.18653/v1/D19-5808. url : https://aclanthology.org/D19-5808.

Liu, Yinhan et al. (juill. 2019). “RoBERTa : A Robustly Optimized BERT Pretraining Approach”. In : url : http://arxiv.org/abs/1907.11692.

Loginov, Vladimir (2021). “Why you should try the real data for the scene text recognition”. In : *arXiv preprint arXiv :2107.13938*.

Lu, Dengsheng et Qihao Weng (2007). “A survey of image classification methods and techniques for improving classification performance”. In : *International journal of Remote sensing* 28.5, p. 823-870.

Marti, U.-V. et H. Bunke (1999). “A full English sentence database for off-line handwriting recognition”. In : *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*, p. 705-708. doi : 10.1109/ICDAR.1999.791885.

Martin, Louis et al. (juill. 2020). “CamemBERT : a Tasty French Language Model”. In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online : Association for Computational Linguistics, p. 7203-7219. doi : 10.18653/v1/2020.acl-main.645. url : https://aclanthology.org/2020.acl-main.645.

Masoudnia, Saeed et Reza Ebrahimpour (2014). “Mixture of experts : a literature survey”. In : *Artificial Intelligence Review* 42.2, p. 275-293.

Mehrabi, Ninareh et al. (2021). “A survey on bias and fairness in machine learning”. In : *ACM Computing Surveys (CSUR)* 54.6, p. 1-35.

Mihaylov, Todor et al. (oct. 2018). “Can a Suit of Armor Conduct Electricity ? A New Dataset for Open Book Question Answering”. In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium : Association for Computational Linguistics, p. 2381-2391. doi : 10.18653/v1/D18-1260. url : https://aclanthology.org/D18-1260.

Mishra, A., K. Alahari et C. V. Jawahar (2012). “Scene Text Recognition using Higher Order Language Priors”. In : *BMVC*.

Nallapati, Ramesh, Bing Xiang et Bowen Zhou (2016). “Sequence-to-Sequence RNNs for Text Summarization”. In : *CoRR* abs/1602.06023.

Napoles, Courtney, Matthew Gormley et Benjamin Van Durme (juin 2012). “Annotated Gigaword”. In : *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*. Montréal, Canada : Association for Computational Linguistics, p. 95-100. url : https://aclanthology.org/W12-3018.

Olteanu, Alexandra et al. (2019). “Social data : Biases, methodological pitfalls, and ethical boundaries”. In : *Frontiers in Big Data* 2, p. 13.

Papineni, Kishore et al. (juill. 2002). “Bleu : a Method for Automatic Evaluation of Machine Translation”. In : *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA : Association for Computational Linguistics, p. 311-318. DOI : 10.3115/1073083.1073135. URL : https://aclanthology.org/P02-1040.

Poliak, Adam et al. (juin 2018). “Hypothesis Only Baselines in Natural Language Inference”. In : *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana : Association for Computational Linguistics, p. 180-191. DOI : 10.18653/v1/S18-2023. URL : https://aclanthology.org/S18-2023.

Prates, Marcelo OR, Pedro H Avelar et Luis C Lamb (2020). “Assessing gender bias in machine translation : a case study with google translate”. In : *Neural Computing and Applications* 32.10, p. 6363-6381.

Qi, Peng et al. (2020). “Stanza : A Python Natural Language Processing Toolkit for Many Human Languages”. In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*. URL : https://nlp.stanford.edu/pubs/qi2020stanza.pdf.

Qu, Yingqi et al. (juin 2021). “RocketQA : An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering”. In : *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. Online : Association for Computational Linguistics, p. 5835-5847. DOI : 10.18653/v1/2021.naacl-main.466. URL : https://aclanthology.org/2021.naacl-main.466.

Radford, Alec et Karthik Narasimhan (2018). “Improving Language Understanding by Generative Pre-Training”. In.

Radford, Alec, Jeff Wu et al. (2019). “Language Models are Unsupervised Multitask Learners”. In.

Raffel, Colin et al. (2020). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In : *Journal of Machine Learning Research* 21.140, p. 1-67. URL : http://jmlr.org/papers/v21/20-074.html.

Rajpurkar, Pranav, Robin Jia et Percy Liang (juill. 2018). “Know What You Don’t Know : Unanswerable Questions for SQuAD”. In : *Proceedings of the 56th Annual Meeting*

of the Association for Computational Linguistics (Volume 2 : Short Papers). Melbourne, Australia : Association for Computational Linguistics, p. 784-789. DOI : 10.18653/v1/P18-2124. URL : https://aclanthology.org/P18-2124.

RAJPURKAR, Pranav, Jian ZHANG et al. (nov. 2016). "SQuAD : 100,000+ Questions for Machine Comprehension of Text". In : *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas : Association for Computational Linguistics, p. 2383-2392. DOI : 10.18653/v1/D16-1264. URL : https://aclanthology.org/D16-1264.

REDDY, Siva, Danqi CHEN et Christopher D. MANNING (2019). "CoQA : A Conversational Question Answering Challenge". In : *Transactions of the Association for Computational Linguistics* 7, p. 249-266. DOI : 10.1162/tacl_a_00266. URL : https://aclanthology.org/Q19-1016.

REIMERS, Nils et Iryna GUREVYCH (nov. 2019). "Sentence-BERT : Sentence Embeddings using Siamese BERT-Networks". 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)*. Hong Kong, China : Association for Computational Linguistics, p. 3982-3992. DOI : 10.18653/v1/D19-1410. URL : https://aclanthology.org/D19-1410.

REN, Siyu et Kenny Q ZHU (2021). "Knowledge-driven distractor generation for cloze-style multiple choice questions". In : *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 35. 5, p. 4339-4347.

RIBEIRO, Marco Tulio, Sameer SINGH et Carlos GUESTRIN (2016). "" Why should i trust you ?" Explaining the predictions of any classifier". In : *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, p. 1135-1144.

RICHARDSON, Matthew, Christopher J.C. BURGES et Erin RENSHAW (oct. 2013). "MCTest : A Challenge Dataset for the Open-Domain Machine Comprehension of Text". In : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA : Association for Computational Linguistics, p. 193-203. URL : https://www.aclweb.org/anthology/D13-1020.

RUSH, Alexander M., Sumit CHOPRA et Jason WESTON (sept. 2015). "A Neural Attention Model for Abstractive Sentence Summarization". In : *Proceedings of the 2015 Conference*

on Empirical Methods in Natural Language Processing. Lisbon, Portugal : Association for Computational Linguistics, p. 379-389. DOI : 10.18653/v1/D15-1044. URL : https://aclanthology.org/D15-1044.

SENNRICH, Rico, Barry HADDOW et Alexandra BIRCH (août 2016). "Neural Machine Translation of Rare Words with Subword Units". In : *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. Berlin, Germany : Association for Computational Linguistics, p. 1715-1725. DOI : 10.18653/v1/P16-1162. URL : https://aclanthology.org/P16-1162.

SIMOULIN, Antoine et Benoit CRABBÉ (2021). "Un modèle Transformer Génératif Préentrainé pour le_ français (Generative Pre-trained Transformer in_ (French) We introduce a French adaptation from the well-known GPT model)". In : *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale*, p. 246-255.

SINGH, Krishna Kumar et al. (juin 2020). "Don't Judge an Object by Its Context : Learning to Overcome Contextual Bias". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

STANOVSKY, Gabriel, Noah A. SMITH et Luke ZETTLEMOYER (juill. 2019). "Evaluating Gender Bias in Machine Translation". In : *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy : Association for Computational Linguistics, p. 1679-1684. DOI : 10.18653/v1/P19-1164. URL : https://aclanthology.org/P19-1164.

SURESH, Harini et John V GUTTAG (2019). "A framework for understanding unintended consequences of machine learning". In : *arXiv preprint arXiv :1901.10002* 2, p. 8.

TAFJORD, Oyvind et Peter CLARK (2021). "General-Purpose Question-Answering with Macaw". In : *ArXiv* abs/2109.02593.

TALMOR, Alon et al. (juin 2019). "CommonsenseQA : A Question Answering Challenge Targeting Commonsense Knowledge". 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)*. Minneapolis, Minnesota : Association for Computational Linguistics, p. 4149-4158. DOI : 10.18653/v1/N19-1421. URL : https://aclanthology.org/N19-1421.

Trischler, Adam et al. (août 2017). “NewsQA : A Machine Comprehension Dataset”. In : *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada : Association for Computational Linguistics, p. 191-200. doi : 10.18653/v1/W17-2623. url : https://aclanthology.org/W17-2623.

Tsuchiya, Masatoshi (mai 2018). “Performance Impact Caused by Hidden Bias of Training Data for Recognizing Textual Entailment”. In : *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan : European Language Resources Association (ELRA). url : https://aclanthology.org/L18-1239.

Van Aken, Betty et al. (2019). “How does bert answer questions ? a layer-wise analysis of transformer representations”. In : *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, p. 1823-1832.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In : *Advances in Neural Information Processing Systems*. Sous la dir. d’I. Guyon et al. T. 30. Curran Associates, Inc. url : https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vincent, Pascal et al. (2008). *Extracting and Composing Robust Features with Denoising Autoencoders*. url : http://www.iro.umontreal.ca/%C3%A2%CB%86%C2%BClisa.

Wang, Kai et Serge Belongie (2010). “Word Spotting in the Wild”. In : *Computer Vision – ECCV 2010*. Sous la dir. de Kostas Daniilidis, Petros Maragos et Nikos Paragios. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 591-604. isbn : 978-3-642-15549-9.

Welbl, Johannes, Nelson F. Liu et Matt Gardner (sept. 2017). “Crowdsourcing Multiple Choice Science Questions”. In : *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark : Association for Computational Linguistics, p. 94-106. doi : 10.18653/v1/W17-4413. url : https://www.aclweb.org/anthology/W17-4413.

Wolf, Thomas et al. (oct. 2020). “Transformers : State-of-the-Art Natural Language Processing”. In : *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*. Online : Association for Computational Linguistics, p. 38-45. url : https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Wu, Yonghui et al. (2016). "Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation". In : *CoRR* abs/1609.08144. arXiv : 1609.08144. URL : http://arxiv.org/abs/1609.08144.

Yang, Zhilin, Zihang Dai et al. (2019). "XLNet : Generalized Autoregressive Pretraining for Language Understanding". In : *Advances in Neural Information Processing Systems*. Sous la dir. de H. Wallach et al. T. 32. Curran Associates, Inc. URL : https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Yang, Zhilin, Peng Qi et al. (2018). "HotpotQA : A Dataset for Diverse, Explainable Multi-hop Question Answering". In : *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yu, Deli et al. (2020). "Towards accurate scene text recognition with semantic reasoning networks". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 12113-12122.

Yuksel, Seniha Esen, Joseph N Wilson et Paul D Gader (2012). "Twenty years of mixture of experts". In : *IEEE transactions on neural networks and learning systems* 23.8, p. 1177-1193.

Zhou, Qingyu et al. (juill. 2017). "Selective Encoding for Abstractive Sentence Summarization". In : *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. Vancouver, Canada : Association for Computational Linguistics, p. 1095-1104. DOI : 10.18653/v1/P17-1101. URL : https://aclanthology.org/P17-1101.

Zhu, Chen et al. (2020). "FreeLB : Enhanced Adversarial Training for Natural Language Understanding". In : *International Conference on Learning Representations*. URL : https://openreview.net/forum?id=BygzbyHFvB.

Zhu, Yukun et al. (2015). "Aligning books and movies : Towards story-like visual explanations by watching movies and reading books". In : *Proceedings of the IEEE international conference on computer vision*, p. 19-27.

Zhu, Zhuotun, Lingxi Xie et Alan Yuille (2017). "Object recognition with and without objects". In : *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, p. 3609-3615.