Approches neuronales pour le résumé abstractif de transcriptions de parole

Paul Tardy

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

Le Mans Université

THÈSE DE DOCTORAT DE

LE MANS UNIVERSITÉ
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par
Paul TARDY

« Approches neuronales pour le résumé abstractif de transcriptions de parole »

Thèse présentée et soutenue à Le Mans Université, LIUM, le 12/07/2021
Unité de recherche : Laboratoire d'Informatique de l'Université du Mans (LIUM)
Thèse N° : 2021LEMA1022

Rapporteurs avant soutenance :

Sophie Rosset	Directrice de Recherche, LISN
Alexandre Allauzen	Professeur, LAMSADE

Composition du Jury :

Président :	Sylvain Meignier	Professeur, LIUM
Examinateurs :	Alexis Nasr	Professeur, LIS
Dir. de thèse :	Yannick Estève	Professeur, LIA
Co-dir. de thèse :	David Janiszek	Maître de Conférence, Paris Université

Invité(s) :

Vincent Nguyen Président, Ubiqus Labs

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

Le Mans Université

Titre : Approches Neuronales pour le résumé abstractif de transcriptions de parole

Mot clés : Résumé Abstractif ; Apprentissage auto-supervisé ; Génération de comptes rendus de réunion ; Résumé inverse ; Réseaux de Neurones ; Tokens de contrôle ; Alignement Automatique ; Évaluation humaine

Résumé : Nous étudions, dans cette thèse, l'application des approches neuronales d'apprentissage profond pour le résumé abstractif de transcription de parole dans le cadre de la génération de comptes rendus de réunions. Ce travail prend place dans un contexte où l'apprentissage profond est omniprésent dans le domaine du Traitement Automatique du Langage Naturel (TALN). En effet, les modèles neuronaux constituent désormais l'état de l'art sur différentes tâches de génération de texte telles que la traduction automatique et le résumé abstractif. Toutefois, l'application du résumé automatique pour la génération de comptes rendus de réunions en français reste très peu explorée. En effet, cette tâche souffre du manque de données disponibles du fait des difficultés à collecter et à annoter de telles données.

Dans ce contexte, notre première contribution consiste en la constitution d'un jeu de données pour cette tâche en alignant des comptes rendus avec les transcriptions automatiques des enregistrements audio de la réunion. Nous proposons une méthodologie associant l'alignement automatique à l'alignement humain. Cette méthodologie nous permet de développer des modèles d'alignement automatique grâce à la constitution de jeux de données d'évaluation tout en facilitant la tâche aux annotateurs humains grâce à l'usage de pré-alignements automatiques.

Ensuite, afin de s'abstraire des contraintes liées à l'annotation – même automatique – nous proposons un pré-entraînement auto-supervisé des modèles afin de tirer profit de grands ensembles de données non-alignées. De plus, nous introduisons le *résumé inverse* nous permettant de générer des données synthétiques et de former des paires d'entraînement à partir de comptes rendus non-alignés. Nous combinons enfin ces deux approches et montrons leur bonne synergie.

Les travaux de cette thèse se concentrent sur l'approche *abstractive* du résumé automatique qui consiste à générer un résumé de toutes pièces, par opposition à l'approche *extractive* où des portions du document sont sélectionnées en tant que résumé. En effet, la rédaction de comptes rendus de réunion à partir de transcriptions automatiques nécessite de reformuler les propos, d'éventuellement les corriger ou de les réorganiser afin de passer d'un langage oral hasardeux à un langage écrit et bien structuré. Pour autant, même les modèles dits *abstractifs* présentent un *biais d'extractivité* consistant à trop copier des mots issus de la source. Afin de limiter ce biais, nous introduisons l'apprentissage explicite du taux de copie attendu au sein du résumé grâce à des tokens de contrôle.

Enfin, nous clôturons ce travail de thèse par une évaluation humaine des comptes rendus automatiques. Cette évaluation nous permet notamment de porter un regard critique sur les performances de nos modèles ainsi que sur le cadre expérimental, notamment sur les métriques et les données utilisées lors de l'évaluation.

Title: Neural Approaches for Abstractive Summarization of Speech Transcription.

Keywords: Abstractive Summarization ; Self-Supervised Learning ; Meeting Report Generation ; Back-summarization ; Neural Networks ; Control Tokens ; Automatic Alignment ; Human Evaluation

Abstract: In this thesis, we study the application of Deep Learning Neural Approaches for abstractive summarization for meetings reports generation.

This work takes place in a context where Deep Learning is omnipresent in the Natural Language Processing field (NLP). In fact, neural models constitute the current state-of-the-art in different language generation tasks such as Machine Translation and Abstractive Summarization.

However, the application of automatic summarization for meeting report generation in french remains unexplored. Indeed, this task suffers from a lack of available data because of difficulties to collect and annotate such data.

In this context, our first contribution consists of the creation of a dataset for this task by aligning meeting reports with automatic transcriptions of the meeting's audio recording. We propose a methodology associating automatic alignment with human alignment.

This methodology enables us to develop automatic alignment models thanks to the annotation of an evaluation dataset while facilitating the human annotation task thanks to the use of automatic pre-alignments.

Then, in order to avoid constraints from the annotation – even automatic – we suggest running a self-supervised pre-training in order to take profit from large amounts of unaligned data. Moreover, we introduce *back-summarization* that allows us to generate synthetic data and create training pairs from unaligned meeting reports. We also combine those two approaches and show their synergy.

In this thesis, we focus our work on the *abstractive* approach of automatic summarization which consists in generating a summary from scratch, as opposed to the *extractive* approach where parts of the source document are selected to form the summary.

Indeed, writing meeting reports from automatic transcriptions requires rephrasing what is being said, optionally correcting it or reorganizing it in order to go from a spoken language to a written, and more formal language.

In order to alleviate this bias, we introduce the explicit learning of the expected copy rate with control tokens.

Finally, we conclude this thesis work with a human evaluation of automatic reports. This evaluation allows us to give a critical look at our models' performances as well as our experimental setup in particular on the metrics and the data used during evaluation.

Remerciements

Ce travail de thèse a été long et difficile et son aboutissement tient en grande partie à l'implication de la part de mes encadrants, du bon cadre de travail dans lequel j'ai pu évoluer et du soutiens de mes proches.

Je tiens à remercier mon directeur de thèse, Yannick Estève, et mon co-encadrant David Janiszek pour leur suivi et leur soutiens tout au long de cette thèse malgré la distance de nos lieux de travail. Merci pour le temps consacré à nos appels réguliers ainsi qu'aux relectures de papiers et du manuscrit.

Je remercie sincèrement Vincent Nguyen, Président d'Ubiqus, avec qui nous avons initié ce projet ambitieux de thèse CIFRE. J'ai pu participé aux débuts de la Recherche et Développement chez Ubiqus Labs dans un cadre idéal pour effectuer mes recherches. Je remercie également mes collègues de l'équipe R&D François, Louis, Linxiao et Valentin mais aussi Frédéric, Nicolas, Guillaume du 14$^{\text{ème}}$ étage ainsi que Grégoire, Fabien, Claire et Florence d'Ubiqus France.

Merci également à Sophie Rosset, directrice de recherche au LISN et Alexandre Allauzen, Professeur au LAMSADE d'avoir accepté d'être les rapporteurs de cette thèse, ainsi qu'à Sylvain Meignier, Professeur au LIUM, et Alexis Nasr, Professeur au LIS, d'avoir accepté de juger ce travail en intégrant le jury lors de la soutenance de cette thèse.

Mes remerciements vont ensuite à toutes les personnes ayant participé à mon éducation, en premier lieux mes parents qui ont su créer un cadre idéal pour mon épanouissement intellectuel et personnel, ainsi qu'à mes professeurs successifs pour avoir su créer ou entretenir l'envie de poursuivre mes études jusqu'en doctorat.

Enfin, je remercie chaleureusement ma famille, mes amis et ma copine pour m'avoir épaulé, soutenu et pour le cadre bienveillant qu'ils forment. L'aboutissement de cette thèse n'aurait pu être possible sans cet équilibre entre travail de recherche et vie personnelle.

Table des matières

Table des figures

Liste des tableaux

Acronymes

AE auto-encodeur

AR auto-régressif

ASR Reconnaissance de la Parole (en anglais Automatic Speech Recognition)

BWS Best-Worst Scaling

GRU Gated Reccurent Unit

IA Intelligence Artificielle

LSTM Long Short Term Memory

RNN Réseau de Neurones Récurrent (en anglais Recurrent Neural Networks)

SHR Split Half Reliability

TALN Traitement Automatique du Langage Naturel

Glossaire

Apprentissage Auto-Supervisé L'Apprentissage Auto-Supervisé consiste à entraîner un modèle à partir de données non alignées (par opposition aux paires d'entraînement de l'apprentissage supervisé) sur une tâche *proxy* comme la reconstitution à partir de données bruitées . 27

Apprentissage Profond Apprentissage Automatique avec un haut niveau d'abstraction (anglais Deep Learning). 23, 47, 157

Apprentissage Supervisé L'Apprentissage Supervisé consiste à entraîner un modèle à partir de paires d'entraînement constituée d'entrées du modèle (aussi appelées *sources*) ainsi que des sorties souhaitées (appelées *cibles*) . 20, 27

Biais d'Extractivité Biais constaté au sein des modèles de résumé automatique en faveur d'un comportement extractif (*cf.* section 3.6) . 87, 88, 117, 122, 137, 139, 149, 150

Encodeur-Décodeur Modèle Neuronal de prédiction de séquence cible (au sein du décodeur) à partir de la représentation d'une séquence source (avec l'encodeur) pour la traduction ou le résumé automatique [Cho *et al.*, 2014]. 21, 60, 61, 63, 67, 69, 70, 73, 88

Mécanisme d'Attention Le mécanisme d'attention est une couche neuronale proposée par [Bahdanau *et al.*, 2014] permettant au modèle d'apprendre à pondérer les entrées en fonction de leur importance à un instant donné (*cf.* section 2.3.2) . 61, 63, 67, 69, 70

Mécanisme de Copie Le mécanisme de copie (aussi appelé Pointeur-Générateur) est une couche neuronale permettant au modèle d'apprendre à copier les mots de le entrées à la place de générer un mot (*cf.* section 3.2.3) . 70, 72, 87, 88, 104, 117, 118

Résumé Inverse Technique d'augmentation des données par l'apprentissage de la tâche inverse du résumé : la génération des *sources* à partir des *cibles* (chapitre 5 Tardy *et al.*, 2020b) . 27, 111, 112, 115, 117, 121

Traduction Neuronale Domaine d'étude de traduction automatique reposant sur les réseaux de neurones (anglais Neural Machine Translation (NMT). 73

Transformer Évolution du modèle encodeur-décodeur basé sur le mécanisme d'attention plutôt que sur des Réseau de Neurones Récurrent (RNN, Recurrent Neural Networks) [Vaswani *et al.*, 2017]. 59, 66, 67, 73, 88, 104, 113, 117, 118, 127

Introduction

Ces dix dernières années ont marqué un tournant dans le domaine de l'Intelligence Artificielle (IA) grâce à la montée en puissance des approches neuronales et de l'apprentissage profond (*Deep Learning*).

Aujourd'hui, le Deep Learning équipe nos smartphones, nos voitures, nous permet de traduire des textes d'une langue à une autre, de reconnaître ou de synthétiser des images et des voix ou d'interagir avec des moteurs de recherche plus pertinents et personnalisés.

Si les principes fondateurs des réseaux de neurones datent du siècle dernier – comme le perceptron multi-couches [Rosenblatt, 1958] ou la retropropagation du gradient d'erreur [Rumelhart *et al.*, 1986] – l'accroissement récente des capacités de calculs (notamment grâce à l'introduction des GPGPU [1]) et du volume de données accessibles sur Internet ont permis des les exploiter à leur plein potentiel. Les approches neuronales ont alors rencontré de nombreux succès dans divers domaines tels que la reconnaissance d'image, la reconnaissance de la parole, la traduction automatique ou la maîtrise de jeux (*e.g.* echecs, go, shogi *etc.*).

Le Deep Learning consiste à représenter une tâche complexe par une hiérarchie de concepts plus simples, organisés en niveaux d'abstractions, répartis en *couches* au sein de réseaux de neurones. L'ensemble des couches sont ensuite conjointement entraînées à partir de très grands ensembles de données : de l'ordre de millions, ou milliards d'exemples. Cela s'oppose notamment aux approches par règles, où les chercheurs doivent concevoir minutieusement l'algorithme pour qu'il donne les bons résultats mais aussi aux approches statistiques qui nécessitent de définir les représentations des données (*feature engineering*).

Dans le cadre du Traitement Automatique de la Parole, les travaux de [Bengio *et al.*, 2000] ont montré l'efficacité des réseaux de neurones pour apprendre conjointement à projeter les mots dans un espace vectoriel et à modéliser la probabilité d'apparition d'un mot à partir des représentations des mots précédents. Ce travail a ouvert la voie, d'une part, à l'apprentissage de relations sémantiques grâce aux vecteurs de mots [Mikolov *et al.*, 2013a] et d'autre part aux approches neuronales pour la génération de texte [Sutskever *et al.*, 2011 ; Graves, 2013]. En particulier, l'architecture *encodeur-décodeur* [Bahdanau

1. Le *General-Purpose computing on Graphic Processing Unit* (GPGPU) désigne l'usage de cartes graphiques (GPU) pour le calcul générique au lieu du processeur (CPU). En effet, les GPU sont plus adaptés au calcul parallèle et peuvent être de plusieurs ordres de grandeurs plus performants qu'un CPU.

et al., 2014 ; Cho *et al.*, 2014 ; Sutskever *et al.*, 2014] étend ce travail à la modélisation de tâche *sequence-to-sequence* c'est-à-dire à la prédiction d'un texte (une séquence de mots) à partir d'un autre texte. Ce modèle, développé dans le cadre de la traduction neuronale a ensuite été appliqué au résumé automatique [Rush *et al.*, 2015 ; Chopra *et al.*, 2016].

Dans ce contexte, nous explorons l'application des réseaux de neurones profonds pour la génération automatique de comptes rendus de réunion, c'est-à-dire le résumé abstractif de transcriptions de parole issues de réunions. Le résumé est dit abstractif s'il comporte des reformulations, des mots nouveaux (par rapport à la source), par opposition au résumé extractif qui se contente de sélectionner des portions du texte source. Cette approche est plus générale et plus proche de la manière de procéder d'un rédacteur : un compte rendu n'est pas un ensemble de citations de la réunion, c'est un document rédigé qui réorganise, corrige et synthétise les propos tenus.

De manière plus générale, les résumés sont utilisés afin de donner un aperçu d'un contenu, d'en décrire les enjeux ou les informations clés et/ou de susciter l'intérêt pour le document. Les enjeux du résumé varient en fonction de sa nature : la quatrième de couverture d'un livre doit éveiller la curiosité sans trop en dévoiler sur l'issue ; le résumé d'un article scientifique doit au contraire évoquer les résultats principaux et les méthodes utilisées. De même, plusieurs critères entrent en compte lors de la rédaction d'un compte rendu, notamment le degré de synthèse à adopter : une transcription mot-à-mot est pertinente dans des contextes critiques où l'on ne peut se permettre de reformuler (*e.g.* cadre juridique) et une synthèse brève permet à une plus large audience de rapidement prendre connaissance des discussions d'une réunion. Le résumé automatique est donc un domaine composé de multiples sous-domaines dont les caractéristiques et les contraintes diffèrent largement.

Problématiques

Dans cette thèse, nous identifions plusieurs verrous scientifiques à la génération automatique de comptes rendus de réunions. D'abord, si le résumé automatique est un sujet de recherche disposant d'une large littérature scientifique, il est très peu abordé sous l'angle du compte rendu de réunion, d'autant moins en français. La plupart des travaux étudient le résumé d'articles de journaux en anglais. Ceci implique notamment :

— que l'on dispose de peu de jeux de données en français qui traitent de comptes

rendus de réunions ;

— que l'on dispose de peu de références en termes d'évaluation, tant en termes de données de test que de résultats comparables ;

— que les résultats obtenus sur d'autres tâches (*i.e.* le résumé de journaux) ne sont pas nécessairement reproductibles : les conclusions tirées quant aux performances des modèles actuels sur d'autres tâches ne sont pas nécessairement applicables au domaine du compte rendu de réunion.

La collecte de données est donc un aspect critique pour aborder ce travail. Or, la collecte en elle même présente des difficultés. Tout d'abord, l'identification de paires *(document ; resumé)* n'est pas triviale car on dispose rarement à la fois du texte original et de son résumé. Par exemple, un rédacteur Ubiqus rédige soit une transcription mot-à-mot soit un compte rendu, rarement les deux. Ensuite, même en disposant de documents et de leurs résumés, il peut être nécessaire de segmenter et d'aligner les paires de segments : les paires originales *(document ; resumé)* peuvent être trop longues pour être utilisées telle quelles par des réseaux de neurones. On procède alors à la segmentation, c'est-à-dire le découpage en segments de textes plus courts puis l'alignement des segments du document avec ceux du résumé. Ceci nécessite de faire appel à des annotateurs ou de développer des méthodes d'annotations automatiques.

La rédaction de compte rendu de réunion nécessite de comprendre les propos prononcés à l'oral pour les retranscrire. Cette retranscription doit gommer les oralités, corriger les fautes éventuelles et globalement améliorer le niveau de langue, la structure et la rigueur de la prise de parole. Ce travail demande par conséquent un bon niveau d'abstractivité : reprendre les propos tels quels, à la manière d'approches extractives, ne serait pas satisfaisant. Beaucoup de travaux sur le résumé abstractif concernent en réalité des tâches où l'abstractivité n'est pas aussi importante et où des modèles relativement extractifs demeurent pertinents. C'est en particulier le cas des jeux de données issus d'articles de journaux dans lesquels les résumés sont parfois essentiellement constitués de phrases issues de l'article. Des travaux ont cherché à augmenter le degré d'abstractivité dans les données afin de mieux évaluer ce critère (par exemple le corpus XSum de Narayan *et al.*, 2020).

Enfin, si l'on souhaite développer des modèles de rédaction automatique de comptes rendus de réunion, il est nécessaire de réfléchir au protocole expérimental permettant de mesurer les progrès. En particulier, il est nécessaire d'avoir des données de référence, c'est-à-dire des paires *(réunion, compte rendu)* (ce que l'on appelle un jeu de test) et une

métrique permettant d'évaluer la qualité du résumé généré. En d'autres termes il faut définir ce qu'est un *bon* compte rendu.

La recherche de métriques d'évaluation des résumés automatiques est un domaine de recherche actif. L'approche retenue alors, la métrique ROUGE [Lin, 2004], s'appuie sur la comparaison lexicale entre le résumé généré et un résumé de référence. Cette métrique pose plusieurs problèmes, notamment son approche lexicale qui pénalise les reformulations (pourtant pertinentes). Ce point est d'autant plus dérangeant qu'il n'existe pas une unique solution : deux rédacteurs pourraient fournir des résumés bien différents et pourtant aussi valide l'un que l'autre. Le score ROUGE de l'un par rapport à l'autre nous indiquerait une différence majeure alors même qu'un évaluateur humain les considérerait équivalent. De manière analogue au domaine de la traduction automatique, qui fait face aux mêmes problématiques [Mathur *et al.*, 2020], les chercheurs en résumé automatique utilisent la métrique ROUGE qui évalue la similarité lexicale avec un résumé de référence. De nombreuses critiques sont adressées face à un tel système :

— il existe une multitude comptes rendus acceptables *i.e.* il n'y a pas une unique solution [2] ;

— la similarité est basée sur les mots (lexicale) et non sur le sens (sémantique) et pénalise donc les reformulations, synthèses et usages de synonymes qui sont pourtant des éléments intéressants au sein d'un résumé ;

— le score est étant calculé comme la similarité par rapport à une référence, le score maximal est atteint pour un résumé identique à la référence. Or, cette référence n'est pas nécessairement parfaite. Par conséquent, un résumé « parfait » obtiendrait un score plus faible que le résumé de référence ce qui n'est pas souhaitable.

Face aux critiques répétées de l'évaluation ROUGE, il est désormais courant de soumettre les résumés automatiques à l'évaluation humaine afin d'appuyer les résultats de l'évaluation automatique. De telles évaluations permettent également ainsi que de livrer un regard critique sur la métrique ROUGE à travers une étude de corrélation entre le score ROUGE et le jugement humain.

2. ROUGE peut prendre en compte plusieurs références ce qui pourrait atténuer ce problème. On constate pourtant que c'est très rarement le cas en pratique : les jeux de données d'évaluations comportent une seule référence par document source.

Contributions et structure du document

Les premiers chapitres de cette thèse sont consacrés à la présentation du sujet et du contexte de la thèse ainsi qu'à l'état de l'art. Dans le **chapitre 1** nous définissons le sujet de la thèse, le résumé automatique et le contexte d'application : le compte rendu de réunion Ubiqus. Le **chapitre 2** introduit l'état de l'art des approches neuronales appliquées au traitement automatique du langage. Dans le **chapitre 3** nous présentons l'état de l'art des approches neuronales pour le résumé automatique de textes.

À partir de l'analyse des verrous scientifiques et de l'étude de l'état de l'art du résumé abstractif appliqué aux comptes rendus de réunion, nous proposons les contributions suivantes.

Le **chapitre 4** traite de la constitution d'un jeu de données pour l'apprentissage supervisé de modèles de génération de comptes rendus abstractifs de réunion à partir de transcriptions automatiques de la parole. Nous présentons une méthodologie pour la segmentation et l'alignement manuel des segments ainsi que le développement de modèles d'alignement automatique. Cette méthodologie nous permet d'obtenir des alignements de référence permettant l'évaluation de modèles d'alignement automatique. D'autre part, l'alignement automatique nous permet de fournir un pré-alignement aux annotateurs afin de leur faciliter la tâche.

Dans le **chapitre 5** nous proposons deux approches pour l'apprentissage à partir de données non alignées afin de tirer profit d'une vaste quantité de données en contournant les contraintes de l'alignement de données (même automatique). On explore alors le pré-apprentissage auto-supervisé des modèles à partir de comptes rendus non alignés et nous introduisons le résumé inverse, nous permettant de générer des transcriptions synthétiques à partir de comptes rendus non alignés.

Le **chapitre 6** présente notre approche pour la réduction du biais d'extractivité grâce à des tokens de contrôle. On constate en effet que les modèles de résumé automatique tendent à trop s'appuyer sur le document source lors de la génération du résumé. On observe alors une forte proportion de mots copiés à partir de la source au sein des résumés, s'apparentant donc à une approche extractive. S'il peut être pertinent d'utiliser les mêmes mots au sein du résumé que dans la source, on attend au sein des résumés générés des taux de copie similaires à ce que l'on peut observer dans les résumés de référence. Ce n'est cependant pas le cas : les résumés générés présentent trop de copie, et trop peu d'adaptation : ils ne parviennent pas à déterminer correctement dans quelle mesure il est

pertinent de copier. Nous ajoutons donc à la tâche de génération de résumé l'information du taux de copie souhaité sous forme de valeur discrète. Le modèle est ensuite entraîné à tirer profit de cette information lorsqu'on la lui fournit ou à la prédire sinon.

Enfin, nous exposons dans le **chapitre 7** l'évaluation humaine de comptes rendus de réunion que nous avons menée. Nous proposons une méthodologie d'évaluation reposant sur la comparaison de comptes rendus issus de modèles automatiques et de rédacteurs professionnels par des annotateurs humains. Nous proposons ainsi un regard critique sur les performances des modèles de comptes rendus automatiques, mais aussi par rapport aux métriques et aux données de références utilisées pour l'évaluation.

L'Annexe A présente les contributions logicielles. En effet, les travaux de cette thèse ont donné lieu à de nombreux développements logiciels dont la contribution à des projets *open-source* (notamment OpenNMT-py[3] [Klein *et al.*, 2017]) ou la publication d'outils autour du domaine du résumé automatique. Ces outils, utilisés par des centaines de chercheurs font partie intégrante des contributions de la thèse.

Enfin, nous présentons en Annexe B des extraits de comptes rendus générés par différents modèles de résumé automatique développés au cours de cette thèse.

3. https://github.com/OpenNMT/OpenNMT-py

Chapitre 1

Le résumé automatique

Résumer, c'est rédiger un texte plus court et plus synthétique capturant les informations principales du document d'origine. Un résumé peut être plus ou moins long, plus ou moins exhaustif en fonction du contexte et de l'objectif de celui-ci. Ainsi, on résumera un livre par quelques paragraphes en quatrième de couverture censés attiser la curiosité du lecteurs mais sans trop en dévoiler sur le contenu. Au contraire, le résumé d'un article scientifique doit couvrir succinctement tous les points importants du papier : des motivations au résultats en passant par les méthodes. Enfin, dans le contexte des articles de journaux, un paragraphe d'introduction peut résumer les informations clés de l'article en quelques phrases.

Nous donnons dans ce chapitre une définition formelle du résumé automatique de texte ainsi qu'une vue d'ensemble des différents contextes d'application explorés dans la littérature scientifique. Nous précisons aussi le contexte de nos travaux, les comptes rendus de réunion, leurs différentes natures et les données dont nous disposons.

Table des matières

1.1 Définition du résumé automatique de texte

Dans sa formulation la plus simple, le résumé de texte automatique consiste à produire un texte plus court à partir d'un document source. La manière de rédiger le résumé d'un document dépend de la nature du document et de l'objectif du résumé. En effet, les différents types de résumés peuvent présenter des caractéristiques et des contraintes très diverses :

— Littérature : La quatrième de couvertures des livres présente généralement généralement un résumé dont l'objectif est de susciter l'intérêt du lecteur sans trop en dévoiler sur l'intrigue. Ce résumé est donc particulièrement court par rapport au document traité et omet volontairement des informations.

— Science : les publications scientifiques commencent par un résumé (l'*abstract*) de l'ordre d'une demi-page. L'auteur prend soin de décrire succinctement les motivations, les méthodes et les résultats du travail mené. Il s'agit ici véritablement d'un condensé de la publication pour en décrire les enjeux et permettre aux chercheurs de rapidement s'en faire une idée.

— Journaux : la tâche du résumé appliqué aux articles de journaux peuvent prendre deux formes : la génération du titre de l'article, ou du paragraphe introductif. Dans les deux cas il s'agit d'aller à l'essentiel en décrivant l'information clé de l'article en une ou plusieurs phrases.

— Réunion : il existe une multitude d'approches pour rendre compte d'une réunion de la transcription mot-à-mot, à la synthèse en passant par un compte rendu exhaustif suivant les objectifs de la réunion et du résumé.

Les travaux sur le résumé de texte automatique se distinguent en deux approches. D'une part, l'approche *extractive*, consiste à résumer un document à partir de fragments de celui-ci. Cela revient à sélectionner (à extraire) les parties du document jugées les plus pertinentes. D'autre part, l'approche dite *abstractive*, consiste à rédiger le résumé de toutes pièces, à partir de l'analyse du document source. Cette approche n'est donc pas limitée au vocabulaire ni aux formulations du document source et permet donc un meilleur niveau de reformulation et de synthèse.

Nous proposons une définition plus formelle, issue de [Rush *et al.*, 2015]. Soit un document source $\mathbf{x} = \{x_1, \ldots, x_M\}$, $x_i \in \mathcal{V}$ de M mots issues du vocabulaire $\mathcal{V}$, le résumé de texte automatique consiste à produire un texte de $N < M$ mots qui maximise la fonction objectif $s : V^M \times V^N \mapsto \mathbb{R}$.

Avec une approche extractive, on produit une sous-séquence de $\mathbf{x}$, comme décrit equation 1.1 :

$$\underset{m\in\{1,\dots,M\}^N}{\arg\max}\ s(\mathbf{x}, \mathbf{x}_{[m_1,\dots,m_N]}) \tag{1.1}$$

avec $\mathbf{x}_{[m_1,\dots,m_N]}$, la séquence $x_{m_1}, \dots, x_{m_N}$.

Avec une approche abstractive, le résumé est une nouvelle séquence de N mots tirés dans $\mathcal{V}$ (eq. 1.2).

$$\underset{\mathbf{y}\in\mathcal{V}^N}{\arg\max}\ s(\mathbf{x}, \mathbf{y}) \tag{1.2}$$

Dans le contexte du compte rendu de réunion, où l'on souhaite retranscrire par écrit des discussions orales, l'approche abstractive semble plus pertinente. En effet, les langages oraux et écrit, sont par nature, très différents. Le rédacteur est chargé de rendre plus formel le document en corrigeant les éventuelles erreurs de langages, et d'en améliorer la lisibilité. Ainsi, il est amené à reformuler les propos, à modifier les tournures (passage du « on » au « nous ») ce qui justifie l'emploi d'une approche abstractive. Aussi, l'approche extractive est particulièrement pertinente lorsque la qualité du document d'origine est convenable (par exemple un article de journal, ou une publication scientifique). Or, le langage oral est souvent de moins bonne qualité que l'écrit. Ne faire que citer des passages de la discussions, tels quels, conduiraient à de mauvais comptes rendus. Enfin, les discussions sont transcrites automatiquement. Ceci introduit des erreurs, qu'il est nécessaire de corriger, ce qui va également dans le sens du résumé abstractif.

1.2 Présentation des données d'Ubiqus

1.2.1 Les comptes rendus Ubiqus

Ubiqus est un prestataire de services linguistiques qui propose à ses clients de la traduction, de l'interprétariat et de la rédaction de comptes rendus et synthèses de réunion. Le métier de la rédaction consiste, pour les rédacteurs d'Ubiqus, à assister aux réunions – c'est-à-dire aux séminaires, colloques, ateliers, comités d'entreprises, conseils d'administrations, *etc.*– à prendre des notes et capturer les discussions afin de les restituer. Les travaux de cette thèse se concentrent l'application du résumé automatique de texte à la rédaction de comptes rendus de réunion.

La forme de cette restitution dépend du contexte, des besoins et des objectifs du

client. En particulier, le client et le rédacteur doivent préalablement définir le niveau d'exhaustivité ou de synthèse qui correspond le mieux à la réunion. En effet, la tâche du rédacteur est de restituer de manière écrite les discussions qui ont eu lieu. Dès lors, il peut être décidé de rester très fidèle aux propos et aux formulations, par exemple dans un contexte très sensible, ou à l'inverse de rédiger synthétiquement les discussions afin de présenter plus clairement les débats. Ce point étant central dans la rédaction de compte rendus et synthèse de réunion, UBIQUS l'a formalisé sous forme de différents *niveaux de retraitement*(voir tableau 1.1). Ainsi, les niveaux de retraitements sont clairement définis, avec leurs objectifs, le contexte correspondant, ainsi que la longueur approximative du document final. La longueur est exprimée en nombre de pages par heure de réunion, une page étant, au sens d'UBIQUS, équivalent 410 mots indépendamment du nombre de pages « visuelles » présentes dans le document.

Dans cette thèse, nous cherchons à appliquer les méthodes de résumé automatiques au domaine du compte rendu de réunion. Nous pouvons considérer que n'importe quelle niveau de retraitement, est une sorte de résumé d'un niveau plus exhaustif (c'est-à-dire plus haut dans le tableau). On note toutefois des différences quant à la nature du résumé. En effet, les *comptes rendus*, visent à reformuler un discours oral en discours écrit. Le discours est reformulé, et de fait, raccourcis en termes de longueur, mais pas compressé en termes d'information. À l'inverse, les *synthèses* comportent une part de compression de l'information. En théorie, nous pourrions appliquer le résumé automatique sur n'importe quelle paire de niveaux de retraitement, en apprenant à générer le plus synthétique à partir du plus exhaustif. En pratique, l'entraînement de systèmes neuronaux nécessite un volume de données important. De plus, dans l'optique de procéder à un entraînement supervisé de réseaux neuronaux, il est nécessaire d'avoir des données alignées, c'est-à-dire des paires de documents tels que l'un est le résumé (compte rendu ou synthèse) de l'autre. Ainsi, ne sont considérable en pratique, que les paires de niveaux de retraitements dont l'on détient assez de données alignées. Il s'agit d'une contrainte forte, car dans la grande majorité des cas, UBIQUS ne rédige qu'un seul document, dans le niveau de retraitement demandé par le client. Ne disposant pas de paires de document, nous choisissons d'appliquer le résumé automatique sur la source directement, c'est-à-dire la réunion, par le biais de son enregistrement. Afin de pouvoir finaliser le document à l'issue de la réunion, les rédacteurs d'UBIQUS, présents sur site, enregistrent les discussions de la réunion en format audio. Nous disposons donc, pour chaque niveau de retraitement, de l'enregistrement ainsi que du document final.

Niveau de retraitement	Nombre de pages (=410 mots) par heure de réunion	Consignes rédactionnelles
Transcription	> 18	Rester fidèle à la forme orale
Compte rendu exhaustif	$[12; 15]$	Aucune sélection de l'information. Débarrasser le discours de toute oralité.
Compte rendu révisé	$[8; 10]$	Éliminer du discours ce qui, justifié à l'oral, devient inutile à l'écrit.
Synthèse standard	≈ 6	Éliminer les informations accessoires ou peu signifiantes.
Synthèse brève	≈ 4	Ne retenir que les informations relevant directement du cœur de la réflexion.
3 pages par heure	≈ 3	Rédiger une synthèse plus courte que la synthèse brève.
Note de synthèse	≈ 2	Ne garder que la substance d'une manifestation.
Flash Horaire	≤ 1	Rédiger un document « très court ».

TABLE 1.1 – Les niveaux de retraitements de l'information des comptes rendus et synthèses de réunion d'UBIQUS

1.2.2 La transcription de la parole

Les transcriptions des réunion sont générées à partir des enregistrements audio grâce au système de Reconnaissance de la Parole (ASR, Automatic Speech Recognition) interne d'Ubiqus.

Le modèle ASR est entraîné grâce aux outils Kaldi [Povey *et al.*, 2011] à partir des enregistrements et transcriptions dont dispose Ubiqus. Les enregistrements audio – long de dizaines de minutes voire de plusieurs heures – sont segmentés en *groupes de souffles* (en anglais *utterances*) – de l'ordre de dizaines de secondes – et alignés aux sections de textes correspondants dans la transcription comme présenté en section 2.1 de [Hernandez *et al.*, 2018]). Le modèle est ensuite entraîné de manière supervisée à partir des paires *(audio, transcription)* pour chaque groupe de souffle.

Les données, représentants 700 heures de réunion, sont ainsi segmentées et alignées en 476k paires d'entraînement. Le modèle acoustique est un *chain model* de Kaldi, avec une architecture TDNN-F [Povey *et al.*, 2018]. En plus du modèle acoustique, nous ajoutons un modèle de langage (*RNNLM*, voir sections 2.2.2 et 2.1.4) similaire à [Sak *et al.*, 2014] pour recalculer les scores des hypothèses issues du modèle acoustique.

Ce système de reconnaissance de la parole est décrit plus en détails dans [Hernandez *et al.*, 2018].

1.2.3 Segmentation : des données brutes à la définition du résumé automatique de réunion

Nos données, au format brut, sont alors des transcriptions de réunion (pouvant durer plusieurs heures) et des comptes rendus, de dizaines de pages. De telles dimensions ne sont pas adaptées à l'entraînement de réseaux de neurones. Il est donc nécessaire, avant cela, de segmenter les données, c'est-à-dire de les diviser en segments plus court, de l'ordre de grandeur d'un paragraphe.

Il n'est pas trivial de trouver une granularité pertinente. En effet, il est nécessaire que les segments ainsi créés aient une certaine unité. L'objectif est de pouvoir segmenter le compte rendu d'une part, et la transcription, d'autre part, de sorte que le segment compte rendu puisse être généré à partir du segment transcription. Cette tâche est difficile par nature, car chaque partie du compte rendu peut nécessiter (i) des connaissances externes à la seule discussion, sur le contexte ou le client, ainsi que (ii) la connaissance du reste de la réunion.

On peut cependant supposer que cette limitation est d'autant plus présente dans les niveaux de retraitements les plus synthétiques. Le choix du compte rendu exhaustif permet donc d'atténuer cette limitation. De plus, par nature, le compte rendu exhaustif garantit que la quasi-totalité des prises de parole des intervenants sont retranscrites, et leur chronologie respectée. Ainsi, nous nous concentrons sur la tâche de génération de comptes rendus *d'interventions*, c'est-à-dire des prises de paroles ininterrompues d'un unique intervenant. En d'autres termes, une réunion a autant *d'interventions* qu'il n'y a de changement d'intervenant. On peut alors considérer un compte rendu comme la somme des comptes rendus de chaque *intervention*.

Segmenter le compte rendu en *interventions* est particulièrement facile : celles-ci sont explicitement délimitées. En effet, chaque *intervention* commence par la mention du nom ou de la fonction de l'interlocuteur. Ces mentions sont associées à un *style*[1] qui leur est propre permettant leur identification sans erreur.

La transcription automatique est quant à elle segmentée par le moteur en *groupes de souffle* distincts, c'est-à-dire en séquences de mots prononcée par un même intervenant, qui commence et termine par une pause. Une intervention comporte un ou plusieurs *groupes de souffles*, eux-mêmes composés d'une ou plusieurs phrases. En l'état, nous ne disposons donc pas de segments de transcriptions correspondants à une *intervention*. *L'alignement* des segments, c'est-à-dire l'association de chaque *intervention* du compte rendu aux *groupes de souffle* correspondants dans la transcription est abordée en chapitre 4.

1.3 Cadre expérimental

L'avancée des travaux de recherche sur le résumé automatique est guidée par l'établissement de cadres expérimentaux communs. En particulier, la mise en commun de jeux de données et de métriques d'évaluation. Dans cette section, nous présentons d'abord les métriques utilisées dans le cadre de l'évaluation des résumés automatiques (section 1.3.1) puis les jeux de données d'apprentissages et de tests utilisés dans la littérature et dans cette thèse (section 1.3.2).

1. Ici, le *style* désigne une méta-donnée associée au texte. Tous les documents Ubiqus utilisent la même méta-donnée, à savoir le style `Nom` qui peut être associé à différentes mises en forme en fonction des clients ou du contexte.

1.3.1 Métriques

Évaluation des résumés automatiques avec ROUGE La métrique la plus couramment utilisée pour l'évaluation de résumé automatique est ROUGE [Lin, 2004]. Cette métrique est similaire à BLEU [Papineni *et al.*, 2002], utilisée en traduction automatique, en cela qu'elle mesure la similarité lexicale entre un texte (le résumé) et sa référence (le résumé attendu). En d'autres termes, ROUGE mesure la proportions de mots en commun au sein du résumé et de la référence. ROUGE mesure également les proportions de suites de n mots que l'on appelle n-grammes. On note ces mesures ROUGE-n où n est le nombres de mots considérés. Ainsi, ROUGE-1 mesure les proportions de mots, ROUGE-2 mesure les proportions de paires de mots (des bigrammes, ou 2-grammes) *etc.*.

Soit D un document de N mots $D = \{w_1, \ldots, w_N\}$, on a l'ensemble des n-grammes de D :

$$\text{gram}_n(D) = \{(w_1, \ldots, w_n), (w_2, \ldots, w_{n+1}), \ldots, (w_{N-n+1}, \ldots, w_N)\} \tag{1.3}$$

En particulier, on peut dénombrer le nombre de n-grammes en commun pour un résumé automatique S et sa référence R :

$$\text{match}_n(S, R) = |\text{gram}_n(S) \cap \text{gram}_n(R)| \tag{1.4}$$

On peut alors choisir de diviser le nombre de n-grammes communs par le nombre de mots dans le résumé automatique – c'est ce qu'on appelle la *précision* (P, eq. 1.6) – ou par le nombre de mots de le résumé de référence – c'est le rappel (R, eq. 1.5). Enfin, la *F-mesure* (F) est la moyenne harmonique entre la *précision* et le *rappel* (1.7)

$$\text{ROUGE-}n(P) = \frac{\text{match}_n(S, R)}{\text{gram}_n(S)} \tag{1.5}$$

$$\text{ROUGE-}n(R) = \frac{\text{match}_n(S, R)}{\text{gram}_n(R)} \tag{1.6}$$

$$\text{ROUGE-}n(F) = 2 \times \frac{\text{ROUGE-}n(P) \times \text{ROUGE-}n(R)}{\text{ROUGE-}n(P) + \text{ROUGE-}n(R)} \tag{1.7}$$

Une autre variante est couramment utilisée : ROUGE-L, qui mesure la longueur de la *plus grande sous-séquence commune*[2]. De même que pour ROUGE-n, on peut utiliser la

2. https://fr.wikipedia.org/wiki/Plus_longue_sous-séquence_commune

précision en divisant par la longueur du résumé automatique ; le rappel en divisant la par longueur du résumé de référence ; ou la *F-mesure*, la moyenne harmonique des deux.

Dans le cadre de l'évaluation de résumés automatiques, les métriques les plus couramment utilisées sont ROUGE-1(F), ROUGE-2(F) et ROUGE-L(F).

Dans nos expériences, nous utilisons `files2rouge`[3] pour le calcul du score ROUGE.

Limites de la métrique ROUGE L'usage de la métrique ROUGE pour l'évaluation de résumé abstractifs est critiqué dans la mesure où ne sont pris en compte que les correspondances lexicales entre les mots. En d'autre terme, un mot qui ne serait pas le même que celui du résumé de référence sera pénalisé, qu'il soit très proche en sens ou au contraire très éloigné. En particulier, ceci pénalise les reformulations ou l'usage de synonymes qui sont pourtant des comportements pertinents au sein d'un résumé abstractif. Ceci est particulièrement problématique du fait qu'il n'existe pas un unique résumé pertinent, mais que plusieurs alternatives sont possibles.

En réponse, la métrique Meteor [Denkowski & Lavie, 2015] propose de prendre en compte les synonymes dans la mesure du score. Pour autant cette métrique ne s'est pas imposé dans la communauté du résumé automatique.

Mesure de l'*abstractivité* des résumés Pour les travaux sur le résumé *abstractif* de résumé, il est pertinent d'évaluer également dans quelles mesures les résumés générés sont effectivement *abstractifs*. En particulier, on peut mesurer la propension du modèle générer de *nouveaux mots* (*i.e.* absent de la source) ou à l'inverse mesurer le *taux de copie*. En particulier, on attend d'un modèle abstractif que son taux de copie soit proche de celui observer au sein des données de références. On peut mesurer le taux de copie avec la métrique ROUGE-n(P) du résumé par rapport au document source (à la place du résumé de référence lors de l'évaluation).

Évaluation Humaine Compte tenu des critiques apportées aux métriques automatiques, il est intéressant de soumettre les méthodes de résumé automatique à l'évaluation humaine.

3. https://github.com/pltrdy/files2rouge

1.3.2 Jeux de données

L'évolution du domaine du résumé automatique s'est faite conjointement avec la constitution de jeux de données partagés par la communauté et servant de repères. Les articles de journaux ont attirés une attention particulière, sûrement du fait de leur abondance et de leur accès public permettant une exploration du web automatique. Dans un premier temps, la recherche s'est concentrée sur le résumé d'une seule phrase, dans le cadre de génération de titres pour des articles de journaux (en anglais *headline generation*). C'est en particulier le cas des tâches partagées DUC-2003 et DUC-2004 [Over *et al.*, 2007] ainsi que Gigaword [Graff *et al.*, 2003 ; Rush *et al.*, 2015].

Ensuite a été abordé le résumé d'articles complets de journaux, en un résumé de quelques phrases, avec notamment les corpus `CNN/DailyMail` [Hermann *et al.*, 2015 ; Nallapati *et al.*, 2016] et `New-York Times` [Sandhaus, 2008]. De plus, [Narayan *et al.*, 2020] ont proposé `XSum` un corpus où l'article – de plusieurs phrases – est résumé en une seule phrase. Les auteurs mettent l'accent sur l'aspect plus abstractif de leur corpus par rapports à `CNN/DailyMail` et `NYT`.

Les articles de la littérature scientifique sont également des candidats intéressants à l'étude du résumé automatique car il est d'usage que les auteurs rédigent un résumé (*abstract*) en début d'article. Des corpus d'articles scientifiques tels que `PubMed` ou `PeerRead` ont ainsi été utilisés dans le cadre du résumé automatique.

Autant que nous sachions, le seul jeu de données de compte rendu de réunions en français est le corpus `AMI`. Celui-ci est composé de transcriptions de réunions et de leur synthèse. Toutefois, ce jeu de données est composé de 118 réunions ce qui limite son intérêt dans le cadre d'entraînement de réseaux de neurones supervisés. Les résumés sont beaucoup plus courts que les entrées (respectivement 281 et 6100 mots) et synthétisent les discussions pour chaque sujets abordés. Les résumés du corpus `AMI` ne sont donc pas directement comparables aux comptes rendu Ubiqus présentés en section 1.2.

La tâche du résumé automatique à aussi été étudié dans le cadre des articles Wikipédia avec la génération du paragraphe d'introductif à partir de l'article et des sources : `WikiSum` de P. J. Liu *et al.*, 2018 ; mais aussi appliqué à la littérature (`Booksum`) et aux résumés multilingues (`MLSum`, Scialom *et al.*, 2020a).

Comme discuté en section précédente (1.1), des résumés de natures différentes présentent des caractéristiques et des contraintes très différentes. Nous présentons alors, dans le tableau 1.2, plusieurs jeux de données issus de la littérature ou de ce travail de thèse (`man`, `auto` et `public_meetings` du chapitre 4 ; `back` et `valid` du chapitre 5).

Nous mettons en avant les différences de caractéristiques à travers différentes mesures :

— **Source des données :** la première ligne précise l'origine des données de résumé. Nous présentons des jeux de données d'articles de journaux (*News*) ; de publications scientifiques (*Science*) ; de rapport de réunions (*Réunion*) ; ainsi que des données d'Ubiqus collectées durant cette thèse.

— **Quantité de données :** les deux lignes suivantes mentionnent le nombres de paires *(texte source ; résumé)* respectivement dans le jeu d'entraînement et celui de test (une données est absente pour les jeux qui sont exclusivement destiné à l'entraînement ou à l'évaluation).

— **Nombre de mots :** Nous rapportons le nombre de mots moyen (noté μ) présents dans la source *src* (et respectivement dans la cible *tgt*) ainsi que le *coefficient de variation*[4] (noté c_v).

— **Nombre de phrases :** Moyenne et coefficient de variation des nombres de phrases au sein du document source et du résumé cible.

— **Compression :** Exprimée – en moyenne et coefficient de variation – comme le pourcentage de compression appliqué au document source en termes de nombres de mots, c'est-à-dire le pourcentage de mots en moins dans le résumé par rapport au document source *i.e.* $compression = 1 - \frac{\#résumé}{\#document}$. Par exemple, un taux de compression de 90% signifierais que le résumé est 10 fois plus court que le document d'origine.

— **Copie :** Les taux de copie au sein des résumés, c'est-à-dire par la proportion de n-grammes du résumé présent dans la source mesuré avec ROUGE. R1 dénote alors le taux de mots du résumé issus de la source, R2 le taux de bigrammes *etc.*.

— **Score des premières phrases**[5] **:** La valeur *lead* correspond au score ROUGE (R1, R2, RL) associé aux premières phrases du document. En effet, il a été constaté qu'une approche aussi naïve que de sélectionner par exemple les 3 premières phrases du document présente de bonnes performances en termes de scores ROUGE. Un score *lead* elevé signifie que les informations les plus pertinentes sont concentrées en début de document. Le nombre de phrases sélectionnées correspond au nombres de phrases du résumé cible.

4. Le coefficient de variation $c_v = \sigma/\mu$ est une mesure de dispersion sans unité. Ce coefficient exprime le degré de variabilité par rapport à la moyenne.

5. https://github.com/pltrdy/lead_sentences

— **Oracle Extractif**[6] **:** Un autre indicateur intéressant est le score qu'obtiendrait un modèle extractif parfait que l'on appelle *Oracle* dans la mesure où le résumé est généré en connaissance de la référence. Le résumé est généré en sélectionnant des phrases de la source qui maximise le score ROUGE par rapport à la référence. Cette mesure nous indique la borne supérieure en termes de score ROUGE atteignable par un modèle extractif.

On constate d'abord que les jeux de données issus d'articles de journaux présentent de forts taux de copie alors même qu'il sont largement utilisés pour l'étude d'approches abstractives. À l'inverse, les données UBIQUS ont un taux de copie et des résultats Ext-Oracle plus faibles (à part `XSum`), notamment pour `man`, `auto` et `valid` (`public_meetings` et `back` présentent des biais expliquant leurs taux de copie plus élevés, présentés respectivement en sections 5.4.3 et 5.4.2).

Ensuite, une différence importante entre les données UBIQUS et les autres jeux de données se situe au niveau du taux de compression. En effet, on observe des taux de compressions autour de 90% sur l'ensemble des autres jeux de données, c'est-à-dire que le résumé est 10 fois moins long que le document d'origine. Les données UBIQUS quant à elles se situent autour de 25 à 33% car la tâche est différentes : il s'agit davantage de restituer les propos que de les résumer au sens strict.

Enfin, une caractéristique intéressante est le coefficient de variation qui mesure la dispersion d'une métrique par rapport à sa moyenne. On constate des coefficients de variations plus élevés que pour les autres jeux de données en particulier au niveau du taux de compression mais aussi au niveau des nombres de mots source et cibles. Ceci traduit la grande hétérogénéité des données UBIQUS qui mêle de grands discours avec de courtes prises de paroles (*e.g.* lors de questions réponses).

1.4 Discussions et conclusion

Nous avons introduit dans ce chapitre la tâche du résumé automatique. Nous avons vu qu'il existe une grande variété de contextes où l'on utilise des résumés, qui ont chacun leurs caractéristiques propres et leurs contraintes. Nous avons présenté les différentes tâches du résumé automatique telles que traitées dans la littérature scientifique en mettant en avant les différences des jeux de données. Enfin, nous nous sommes concentré sur le

6. https://github.com/pltrdy/extoracle_summarization

		CNN/DM	NYT	XSum	PubMed	AMI	man	auto	back	valid	public_meetings
Source		News	News	News	Science	Réunion	Ubiqus	Ubiqus	Ubiqus	Ubiqus	Ubiqus
Paires (train)		287k	589k	204k	21k	98	21k	68k	6.0m	–	–
Paires (test)		11k	33k	11k	2470	20	–	–	–	1000	1068
Phrase src	μ	39.8	24.1	19.1	81.3	568.0	9.1	9.7	10.1	9.4	11.9
Phrase tgt	μ	3.7	1.9	1.0	10.2	18.6	5.7	5.7	4.8	5.9	8.3
Mots src	μ	791.7	550.4	428.6	2312.2	6129.8	172.3	182.5	240.0	178.2	261.0
	c_v	0.5	0.5	0.7	0.5	0.4	1.0	0.9	0.8	1.0	0.9
Mots tgt	μ	55.2	40.7	23.3	263.2	327.6	116.2	114.9	94.5	121.6	181.7
	c_v	0.4	0.7	0.2	0.5	0.3	1.1	1.1	0.6	1.2	0.9
Compr.	μ	91.52	88.28	90.38	86.95	94.03	25.13	27.61	33.23	24.60	28.12
	c_v	0.06	0.19	0.12	0.07	0.03	2.15	2.08	1.84	2.25	1.36
Copie	R1	90.25	84.92	66.63	88.12	83.91	55.45	54.73	63.09	55.38	75.84
	R2	51.28	47.61	17.80	49.76	29.75	27.76	26.42	37.74	27.84	51.90
	R3	30.58	29.01	04.98	28.10	06.45	17.95	16.92	26.42	18.14	39.28
Lead	R1	39.37	25.23	16.65	39.72	17.08	37.71	35.09	43.18	37.24	54.26
	R2	17.46	10.46	01.65	11.73	02.25	18.21	16.09	25.04	17.95	35.33
	RL	35.76	22.46	12.22	36.55	16.07	34.80	32.27	40.51	34.37	51.89
Ext-Oracle	R1	57.09	51.53	28.75	63.43	48.42	47.12	45.66	52.53	46.99	65.31
	R2	33.72	32.79	08.46	33.14	14.33	25.55	24.14	32.68	25.64	45.87
	RL	53.07	47.24	21.03	59.02	45.22	43.49	41.98	49.13	43.34	62.62

Table 1.2 – Présentation de différents jeux de données de résumé automatique

contexte particulier dans lequel s'inscrit cette thèse c'est-à-dire la rédaction automatique de comptes rendus de réunion au sein d'Ubiqus.

Chapitre 2

Approches Neuronales du Traitement Automatique du Langage Naturel

Les approches neuronales sont au coeur de cette thèse. Dans ce chapitre, nous introduisons donc les réseaux de neurones, leur fonctionnement et leurs applications dans le domaine du Traitement Automatique du Langage Naturel.

Table des matières

2.1 Introduction aux réseaux de neurones

2.1.1 Neurone artificiel

Les neurones, en biologie, sont des cellules électriquement excitables qui constituent le système nerveux. Ils sont constitués notamment d'un corps, de dendrites, qui sont les récepteurs de l'influx nerveux (appelé potentiel d'action), et de l'axone qui transmet le potentiel d'action à d'autres cellules. De part et d'autres, les synapses permettent la transmission de l'influx nerveux d'une cellule à l'autre. Les synapses peuvent alors être excitatrice ou inhibitrice. L'influx se propage des dendrites jusqu'au corps du neurone, et si le potentiel post-synaptique est suffisant, il est transmis le long de l'axone.

Inspirés des neurones biologiques, [Mcculloch & Pitts, 1943] ont présenté les neurones artificiels (aussi appelés neurones formels) c'est-à-dire un modèle mathématique où la sortie (similaire à l'axone) est calculé à partir de plusieurs entrées (similaires aux dendrites). Les entrées sont alors pondérées (de même manière que les actions des synapses) puis additionnées (potentiel post-synaptique). La sortie est positive si le signal est supérieur à un seuil donné, nulle sinon.

Plus formellement, soit un neurone à n entrées x_i pour $i \in [\![1, n]\!]$ et une sortie y booléenne (0 ou 1). Le neurone dispose de n poids synaptiques w_i, un par entrée. Le potentiel post-synaptique z est la somme des entrées pondérées par leurs poids synaptiques c'est-à-dire $z = \sum_{i=1}^{n} w_i x_i$. La sortie dépend alors de la valeur du potentiel z et de la valeur du seuil t :

$$y = \varphi(z) = \begin{cases} 0 \text{ si } z \leq t \\ 1 \text{ si } z > t \end{cases} \tag{2.1}$$

Il est cependant plus courant d'utiliser, au lieu du seuil t, un paramètre dit de biais b telle que $b = -t$ et d'utiliser des notations de vecteurs $\boldsymbol{w} = [w_1, \ldots, w_n]^\top$, $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$ par conséquent $\boldsymbol{z} = \boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}^\top \boldsymbol{x}$ où $\cdot$ est le produit scalaire de deux vecteurs. Aussi, on note θ l'ensemble des paramètres considérés, ici $\{\boldsymbol{w}, b\}$ On peut alors ré-écrire 2.1 :

$$y = \varphi(x; \theta) = \begin{cases} 0 \text{ si } \boldsymbol{w}^\top \boldsymbol{x} + b \leq 0 \\ 1 \text{ si } \boldsymbol{w}^\top \boldsymbol{x} + b > 0 \end{cases} \tag{2.2}$$

La valeur de y correspond alors à l'activation du neurone. La fonction φ telle que $y = \varphi(z)$ est alors appelée fonction d'activation. Dans le modèle présenté, il s'agit de la fonction de Heaviside (aussi appelée fonction *en marche d'escalier*) décrite dans 2.1 et

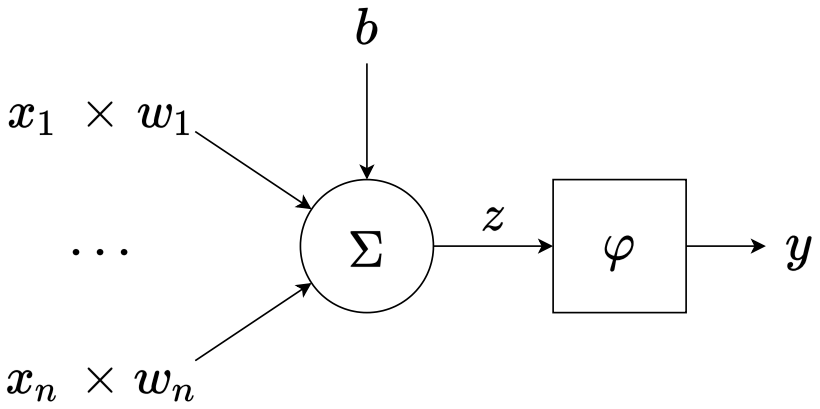


FIGURE 2.1 – Représentation d'un neurone artificiel

2.2.

2.1.2 Apprentissage

L'apprentissage d'une fonction f par un neurone artificiel, consiste à trouver les paramètres θ^* tels que $\varphi(x;\theta^*) = f(x)$. On cherche alors à faire varier les valeurs de θ (à partir d'une initialisation quelconque, éventuellement aléatoire) de manière à se rapprocher de la valeur souhaitée, c'est-à-dire à minimiser la distance $\boldsymbol{\mathcal{L}}(x;\theta) = \varphi(x;\theta) - f(x)$ entre la sortie du neurone et la sortie de la fonction f. Cette distance est aussi appelée l'erreur (ou *loss* en anglais). Intuitivement, on peut imaginer introduire une perturbation des paramètres $\Delta\theta$ de manière à ce que la sortie de notre neurone soit plus proche de la sortie souhaitée c'est-à-dire $\boldsymbol{\mathcal{L}}(x,\theta+\Delta\theta) < \boldsymbol{\mathcal{L}}(x,\theta)$. Il est nécessaire pour cela de déterminer, pour chaque paramètre de θ, la direction à prendre pour minimiser l'erreur, c'est-à-dire s'il faut augmenter la valeur ou la réduire. Ceci correspond à déterminer la *pente* de la fonction erreur au point $(x;\theta)$. Or la pente d'une fonction à un point est donnée par la valeur de sa dérivée, notée $\boldsymbol{\mathcal{L}}'$ ou $\frac{\partial\mathcal{L}}{\partial\theta}$. En d'autres termes, la dérivée nous indique comment modifier θ pour minimiser $\boldsymbol{\mathcal{L}}$ car on a $\boldsymbol{\mathcal{L}}(x;\theta+\Delta\theta) \approx \boldsymbol{\mathcal{L}}(x;\theta) + \Delta\theta\boldsymbol{\mathcal{L}}'(x;\theta)$, au voisinage de θ, c'est-à-dire pour des valeurs de $\Delta\theta$ suffisamment faibles.

En pratique, θ représente un ensemble de paramètres, on calcule alors la dérivée partielle de l'erreur pour chaque paramètre : c'est ce que l'on appelle le gradient de l'erreur que l'on note $\nabla_\theta\boldsymbol{\mathcal{L}}(x;\theta)$. Par exemple, pour un réseau à un seul neurone constitué d'un vecteur de poids $\boldsymbol{w}$ et d'une valeur de biais b on a $\theta = \{\mathbf{w}; b\}$ et $\nabla_\theta\boldsymbol{\mathcal{L}}(x;\theta) = \left[\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}}, \frac{\partial\mathcal{L}}{\partial b}\right]^\top$.

Le gradient de l'erreur nous indique, pour chaque paramètre, la direction à prendre

afin de minimiser la fonction d'erreur. L'apprentissage consiste alors à effectuer des « pas » dans cette direction, c'est-à-dire de mettre à jour des valeurs des paramètres en fonction du gradient : $\theta \leftarrow \theta + \lambda \nabla_\theta \mathcal{L}(x;\theta)$. La taille de ce « pas », appelée le *learning rate* (taux d'apprentissage), est notée λ et constitue un hyper-paramètre[1] important de l'entraînement d'un réseau de neurones.

La fonction d'activation de HEAVISIDE rend cette tâche difficile. En effet, cette fonction est constante sur $]-\infty;t[\cup]t;+\infty[$, sa dérivée est donc nulle. En d'autre terme, il est difficile d'observer l'évolution de l'erreur en fonction des paramètres, et in fine de minimiser cette erreur.

Il est donc préférable d'utiliser d'autres fonction d'activation telles que *sigmoïde* (eq. 2.3), *tangente hyperbolique* (eq. 2.4) ou *ReLU* (eq. 2.5) :

$$\text{Sigmoïde :} \quad \sigma(z) = \frac{1}{1+e^{-z}} \tag{2.3}$$

$$\text{Tangente Hyperbolique :} \quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{2.4}$$

$$\textit{Rectified Linear Unit :} \quad ReLU(z) = max\{0,z\} \tag{2.5}$$

.

2.1.3 Réseaux de neurones et apprentissage profond

Si un neurone seul peut déjà apprendre des fonctions simples, leur réelle force réside dans le fait de pouvoir être connectés entre eux, au sein d'un réseau de neurones.

Dans le cas précédent, nous avions un seul neurone, connecté aux n entrées et à la sortie. Un réseau de neurone consiste à ajouter des neurones, organisés en couches, dites cachées, entre les entrées et les sorties. Chaque neurone d'une couche est connecté en entrée à tous les neurones de la couche précédente, et en sortie à chaque neurone de la couche suivante comme représenté figure 2.3. On dit d'un tel réseau qu'il est *entièrement connecté*.

L'information est propagée, à partir de la couche d'entrée, jusqu'à la couche de sortie en passant par les couches cachées. On parle alors de réseau de neurones à *propagation avant*.

1. Un hyper-paramètre est une variable fixée par l'utilisateur, par opposition aux paramètres θ du réseau. Le nombre de couches, le nombre de neurones sont d'autres hyper-paramètres d'un réseau de neurones.

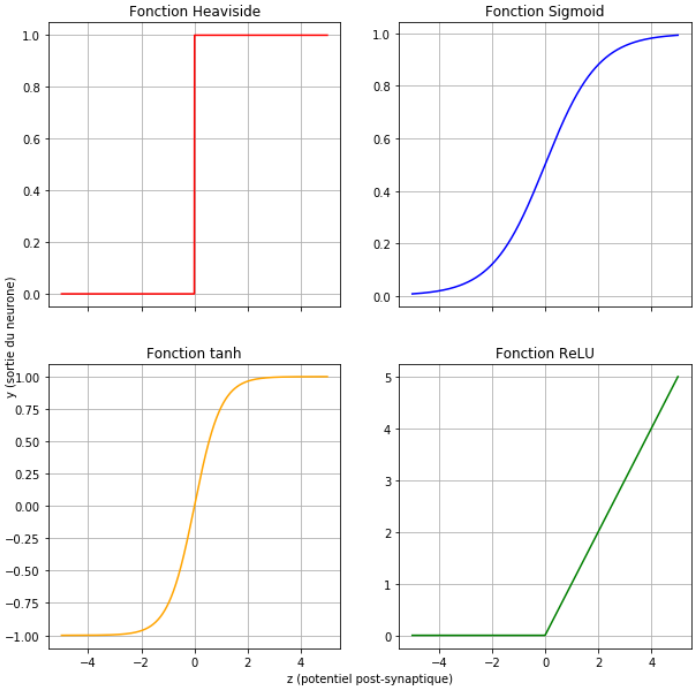


FIGURE 2.2 – Fonctions d'activation : les fonctions HEAVISIDE, sigmoïde, tangente hyperbolique et *rectified linear unit*

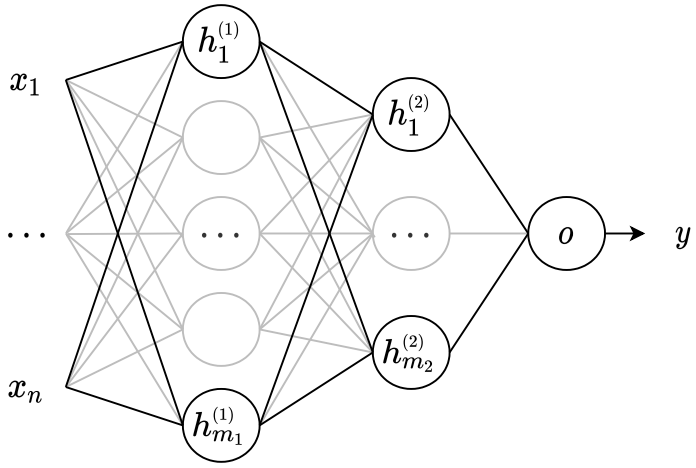


FIGURE 2.3 – Schéma d'un réseau de neurones à n entrées, 2 couches cachées de taille m_1 et m_2 et une sortie

Comme précédemment, chaque neurone pondère ses entrées $\boldsymbol{x}$ par son vecteur de poids $\boldsymbol{w}$. On peut décrire les poids d'une couche par la liste des vecteurs poids et des biais des neurones qui la composent, ou plus simplement, une matrice de poids $\mathbf{W}$ de dimension $n \times m$ (pour une couche de taille m et avec n entrées) et un vecteur de biais $\boldsymbol{b} = [b_1, ..., b_m]$.

Par exemple, le réseau de neurones à deux couches cachées présenté en figure 2.3 est composé des deux matrices de poids $\mathbf{W}_1 \in \mathbb{R}^{n \times m_1}$, $\mathbf{W}_2 \in \mathbb{R}^{m_1 \times m_2}$ et $\mathbf{W}_o \in \mathbb{R}^{m_2 \times 1}$ ainsi que des vecteurs de biais $\boldsymbol{b}_1 \in \mathbb{R}^{m_1}$, $\boldsymbol{b}_2 \in \mathbb{R}^{m_2}$, $\boldsymbol{b}_o \in \mathbb{R}^1$ et implémente la fonction $y = f^{(o)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right) = \varphi^{(o)}\left(\mathbf{W}_o^\top \times \varphi^{(2)}\left(\mathbf{W}_2^\top \times \varphi^{(1)}\left(\mathbf{W}_1^\top \times \boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_o\right)$.

L'intérêt des couches cachées est de combiner plusieurs fonctions non linéaires successivement et ainsi d'apprendre des représentations intermédiaires de différents niveaux d'abstractions. En effet, les différentes couches tendent à se spécialiser par exemple, dans le cas de reconnaissance d'images, d'abord sur les contours de l'image, l'identification des différentes formes avant de projeter ces représentations abstraites, dans les dernières couches, vers une couche de classification, c'est-à-dire d'associer l'image à une classe connue (par exemple le chiffre, l'animal reconnu etc). On dit que la *largeur* d'un réseau est le nombre de neurones au sein de ses couches cachées, et que sa *profondeur* est le nombre de couches. Le terme apprentissage profond (en anglais *deep learning*) vient justement du fait que les architectures modernes tendent à augmenter la *profondeur* du réseau davantage que sa *largeur*.

L'apprentissage des poids d'un réseau de neurones est similaire à l'apprentissage d'un neurone seul tel que décrit en section 2.1.2. L'erreur entre la sortie attendue et la prédiction du réseau est calculée, ainsi que les dérivées partielles pour chaque neurone, c'est ce que l'on appelle la rétropropagation du gradient d'erreur, ou *back-propagation* [Rumelhart *et al.*, 1986] (par opposition à la propagation avant du signal dans le réseau).

Il existe plusieurs algorithmes d'optimisation, c'est-à-dire visant à minimiser la fonction de coût. Le plus commun est sûrement l'algorithme de descente de gradient stochastique (ou SGD pour *Stochastic Gradient Descent*). Plusieurs variantes existent, qui proposent notamment un taux d'apprentissage (λ) différents suivant les paramètres comme avec AdaGrad [Duchi *et al.*, 2010], RMSProp [Hinton *et al.*, 2012] et qui évolue dans le temps (ADAM Kingma et Ba, 2015). Nous ne rentrerons pas ici dans les détails de tels algorithmes.

Durant l'apprentissage, on dispose d'un jeu de données, c'est-à-dire d'un ensemble $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ d'entrées $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ associées à la sortie souhaitée $\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$. Pour chaque paire $(\boldsymbol{x}, \boldsymbol{y})$, on fournit l'entrée $\boldsymbol{x}$ au réseau, on calcule les valeurs de neurones, par couches,

afin d'obtenir la prédiction $\hat{\boldsymbol{y}}$ (propagation avant). On calcule ensuite l'erreur, puis les gradients de l'erreur (par rétropropagation). Enfin les poids du réseau sont mis à jour grâce à l'algorithme d'optimisation.

2.1.4 Réseaux de neurones récurrents

Les réseaux de neurones décrits précédemment fonctionnent par *propagation avant* – c'est-à-dire que l'information se propage de l'entrée vers la sortie. Dans ce contexte, chaque paire d'apprentissage $(\boldsymbol{x}, \boldsymbol{y})$ est traitée indépendamment.

Les RNN quant à eux ont pour objectif de traiter les entrées de façon séquentielle, donc de manière non indépendante. Un réseau de neurones récurrent fonctionne alors de manière itérative, et calcule, à chaque pas t, une sortie $\boldsymbol{y}_t$ ainsi qu'un état $\boldsymbol{h}_t$ qui est transmis au pas $t+1$ (voir fig. 2.4). Ainsi, à chaque itération, le réseau reçoit d'une part l'entrée $\boldsymbol{x}_t$, et d'autre part l'état précédent h_{t-1} et calcule, son nouvel état h_t (équation 2.6) puis, sa sortie y_t (équation 2.7).

$$\boldsymbol{h}_t = \sigma\left(\mathbf{W}_h \boldsymbol{x}_t + \mathbf{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\right) \tag{2.6}$$

$$\boldsymbol{y}_t = \sigma\left(\mathbf{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y\right) \tag{2.7}$$

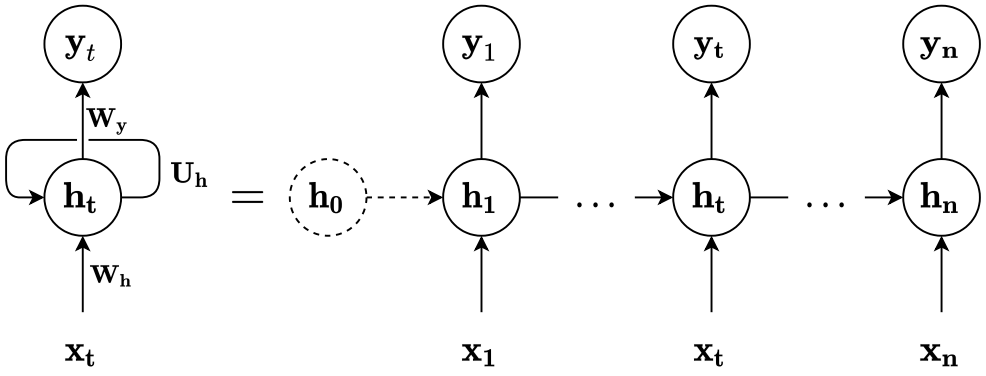


FIGURE 2.4 – Réseau de neurones récurrent. *(À gauche)* représentation du réseau avec ses connexions récurrentes. *(À droite)* représentation du réseau développé dans le temps.

L'intérêt de cette architecture est (i) de traiter des séquences de taille variable et (ii) de partager les paramètres à travers les itérations. En effet, si notre tâche consiste, à l'inverse, à traiter une séquence fixe d'entrées $\boldsymbol{x}_t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\tau\}, \boldsymbol{x}_t \in \mathbb{R}^n$ nous pouvons

utiliser un réseau à propagation avant, dont la couche d'entrée serait de dimension $n \times \tau$. Ceci nécessite que τ soit constant, donc ne permet pas d'entraîner des séquences de taille variable. Le second point est qu'un tel réseau utiliserait des paramètres différents pour chaque élément de la séquence. Au contraire, un RNN traite chaque entrée de la même manière, avec les mêmes paramètres ce qui permet de mettre en commun l'apprentissage.

Ce genre de réseau peut être utilisé dans le traitement automatique du langage afin de traiter des séquences de mots. Dans ce contexte, un RNN permet de prendre des phrases en entrée, de taille variable, et d'apprendre à générer mot à mot une sortie (en fonction de la tâche) et un état (représentant les mots précédents).

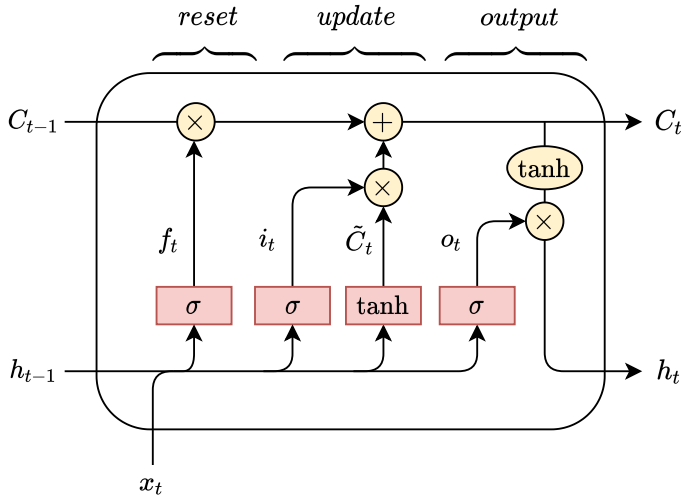
Toutefois, bien que les réseaux récurrents transmettent leur état d'un instant à l'autre, il demeure difficile d'apprendre des relations entre des éléments distants de la séquence, comme discuté dans [Bengio *et al.*, 1994] et [Hochreiter & Schmidhuber, 1997]. Pour pallier ce problème, [Hochreiter & Schmidhuber, 1997] présente une variante de RNN : le Long Short Term Memory (LSTM), qui introduit, en plus de l'état $\boldsymbol{h}_t$ un vecteur de mémoire[2] $\boldsymbol{C}_t$, et plusieurs couches, appelées *portes* (ou en anglais *gate*). Une première porte, appelée *porte d'oubli* ou de *réinitialisation* (en anglais *forget/reset gate*), permet de choisir entre conserver ou oublier ou non certaines valeurs du vecteur mémoire C_t (en multipliant, élément par élément le vecteur mémoire avec un vecteur de valeurs dans $[0; 1]$ issu d'une couche *sigmoïde*). De manière similaire, la seconde porte est constituée d'une couche sigmoïde permettant de sélectionner (en pondérant avec des valeurs dans $[0; 1]$) les éléments de l'état $\boldsymbol{h}_t$ et l'entrée $\boldsymbol{x}_t$ à ajouter à la mémoire : c'est la porte de *mise à jour* (*update gate*). Enfin, une dernière porte (de *sortie*, *output gate*) calcule l'état suivant $\boldsymbol{h}_{t+1}$ à partir de l'état courant $\boldsymbol{h}_t$, de l'entrée $\boldsymbol{x}_t$ et de la mémoire actualisée $\boldsymbol{C}_{t+1}$.

De même, [Cho *et al.*, 2014] présentent leur modèle Gated Reccurent Unit (GRU) dont l'architecture s'inspire du LSTM. On retrouve notamment des portes de *réinitialisation* et de *mise à jour*. Toutefois, ce modèle est légèrement moins complexe que le LSTM en cela qu'il n'utilise que trois portes (au lieu de 4 pour le LSTM) et n'introduit pas de vecteur mémoire.

Les deux architectures – LSTM et GRU – sont présentées respectivement en figure 2.5 et 2.6 ainsi que par les équations 2.8 et 2.9. Dans les équations, $\odot$ désigne le produit de Hadamard[3] et $[\cdot\,;\cdot]$ la concaténation de vecteurs.

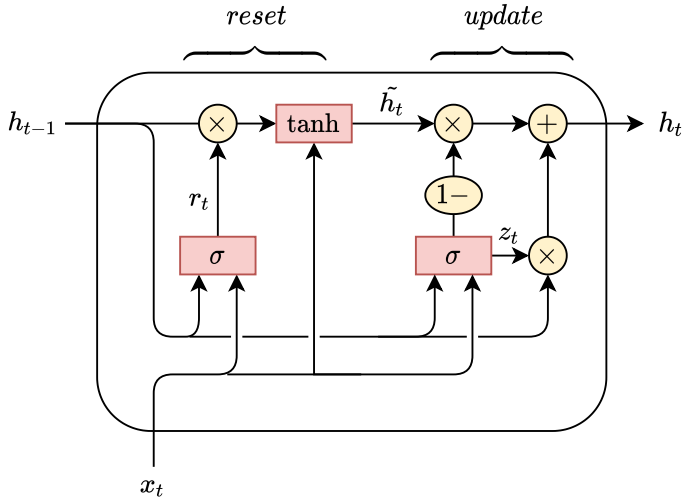
2. À ne pas confondre avec les vecteurs *contexte* $\boldsymbol{c}$ utilisés dans la suite du document.

3. On appelle produit de Hadamard la multiplication terme à terme de vecteur c'est-à-dire, pour deux vecteurs de même dimension n : $\boldsymbol{a} \odot \boldsymbol{b} = (\boldsymbol{a}_1 \times \boldsymbol{b}_1, \ldots, \boldsymbol{a}_n \times \boldsymbol{b}_n)^\top$



$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(\mathbf{W}_f \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\mathbf{W}_i \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i) \\
\boldsymbol{o}_t &= \sigma(\mathbf{W}_o \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{C}}_t &= \tanh(\mathbf{W}_C \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] \\
&\qquad + \boldsymbol{b}_c) \\
\boldsymbol{C}_t &= \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_t \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{C}_t)
\end{aligned}
\tag{2.8}
$$

FIGURE 2.5 – Architecture LSTM



$$
\begin{aligned}
\boldsymbol{z}_t &= \sigma(\mathbf{W}_z \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_z) \\
\boldsymbol{r}_t &= \sigma(\mathbf{W}_r \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_r) \\
\tilde{\boldsymbol{h}}_t &= \tanh(\mathbf{W}_h \cdot [\boldsymbol{x}_t; (\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})] \\
&\qquad + \boldsymbol{b}_h) \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t
\end{aligned}
\tag{2.9}
$$

FIGURE 2.6 – Architecture GRU

2.2 Représentation du langage

Les réseaux de neurones sont avant tout des objets mathématiques qui travaillent avec des vecteurs et des matrices plus qu'avec des mots et des phrases. Ainsi, le traitement automatique du langage naturel via des réseaux de neurones nécessite de représenter numériquement le langage, c'est-à-dire d'associer des nombres, des vecteurs, des matrices aux mots.

2.2.1 Plongement de mots (*word embeddings*)

À première vue, une solution simple pour représenter le langage numériquement consiste à associer chaque mot à un entier, c'est-à-dire en quelque sorte de « numéroter » le vocabulaire. Ainsi, chaque mot w_i du vocabulaire $\mathcal{V}$ est représenté par $num(w_i) = i$, $w_i \in \mathcal{V}$, $i \in [\![1; |\mathcal{V}|]\!]$.

Ces entiers n'ont cependant pas vocation à représenter des valeurs, ni à subir des opérations mathématiques. En effet, additionner, multiplier ou soustraire ces valeurs n'aurait aucun sens. Afin que le réseau de neurone utilise des paramètres spécifiques pour chaque mot sur la couche d'entrée, celle-ci nécessite de disposer d'une dimension par mot du vocabulaire.

On représente ensuite un mot par un vecteur *one-hot* (aussi appelé *un parmi n*) qui vaut 1 sur la dimension de la valeur encodée et 0 sinon, *i.e.* $onehot(w_i) = \left(e_{i,1}, \ldots, e_{i,|\mathcal{V}|}\right)^\top$ avec $e_{i,j} = 1$ si $i = j$, sinon 0.

De cette manière, chaque mot est associé à une dimension et donc à des paramètres spécifiques au sein du réseau de neurones. Par exemple, si l'on a un vocabulaire de n mots alors le mot i est représenté par un vecteur *one-hot* v avec $v_i = 1$ et $j \neq i \Rightarrow v_j = 0$. En utilisant v comme entrée à un réseau de neurone, la couche de paramètres $W \in \mathbb{R}^{n \times d}$ calcule la somme des poids pondérés par le vecteur v *i.e.* $\sum_{j=1}^{n} v_j \times W_j = W_i$ car $v_j = 0 \, \forall j \neq i$. On parvient ainsi à distinguer chaque mot en entrée du réseau et à leur associer des poids différents. De plus, nous obtenons ainsi une représentation dense des mots, c'est-à-dire sous forme de vecteurs réels par opposition à la représentation des vecteurs *one-hot* : discrète et creuse, la représentation étant constituée en immense majorité de 0. Ces vecteurs sont de dimensions bien plus faibles (en général inférieur à 1024 au lieu de plusieurs dizaines de milliers). Ces représentations peuvent être appelées *vecteurs de mots*, *représentations distribuées (ou continues) de mots* ou encore *word embeddings* (*plongement de mots*).

De nombreuses techniques ont été utilisées pour calculer des représentations denses de mots, à commencer par des approches statistiques reposant sur *l'hypothèse distributionnelle* [Firth, 1957] d'après laquelle les mots qui apparaissent dans les mêmes contextes tendent à avoir des sens similaires. L'Analyse Sémantique Latente (LSA, *Latent semantic analysis*) propose de dénombrer les occurrences de chaque mot par document. D'autres approches, comme HAL [Lund & Burgess, 1996] dénombrent des co-occurrences de mots, c'est-à-dire, mesurer le nombre d'apparitions de chaque mot dans le contexte de chaque autre mot. Dans les deux cas, il s'agit de construire une matrice de très grande di-

mension. Ces méthodes utilisent des techniques d'approximation de rang réduit (basée sur la décomposition en valeur singulière (SVD)) afin d'en réduire le nombre de dimensions, et donc la complexité.

Plus récemment, les réseaux de neurones ont été étudiés comme méthode d'apprentissage des plongements de mots. Pour obtenir une représentation dense à partir d'un réseau de neurones, il suffit (théoriquement) que les poids de sa couche d'entrée soient de dimension $|\mathcal{V}| \times d_{emb}$, c'est-à-dire qu'à partir d'une représentation *one-hot* des mots (de dimension $|\mathcal{V}|$, on obtienne une sortie de dimension d_{emb} telle que $d_{emb} \ll |\mathcal{V}|$. L'entraînement de plongements de mots à l'aide de réseau de neurones est présentée dans [Bengio *et al.*, 2000] (approche détaillée en section 2.2.2).

[Collobert & Weston, 2008] ont montré la bonne généralisation des plongements de mots à travers l'apprentissage conjoint de diverses tâches syntactiques comme l'*étiquetage morpho-syntaxique* (*Part-of-Speech Tagging*) et sémantiques comme la *reconnaissance d'entités nommées* (*Named Entity Recognition*) ou l'*étiquetage des rôles sémantiques* (*Semantic Role Labeling*). Ces travaux montrent que des représentation génériques peuvent être apprises et appliquées à des tâches diverses, par opposition aux représentations spécifiques à chaque tâche.

Les plongements de mots ont été popularisés en grande partie grâce aux travaux de Tomas Mikolov, en particulier dans [Mikolov *et al.*, 2013c] où les auteurs présentent un modèle neuronal simple, sans couche cachée et avec une couche de sortie log-linéaire [4] dont l'efficacité permet d'entraîner des plongements sur un très grand nombre de mots (de l'ordre de 10^9). La tâche d'entraînement consistant à prédire un mot w_t à partir de son contexte, formé des k [5] mots précédents et suivants. Les représentations des mots suivants, et précédents, sont additionnées sans notion d'ordre d'où le nom de cette tâche : *Continuous Bag-of-Words* (CBOW). Cette représentation repose sur des vecteurs denses, donc continus, par opposition aux vecteurs discrets tels *one-hot*. De plus, on appelle *sac de mots* (= *bag of words*) un ensemble non ordonné de mots. [Mikolov *et al.*, 2013c] étudient aussi la tâche contraire (appelée *SkipGram*), consistant à prédire le contexte – les k mots précédents et suivants – à partir du mot central.

L'objectif du travail de [Mikolov *et al.*, 2013c] n'est pas tellement d'obtenir un modèle de prédiction du mot manquant ou du contexte, mais surtout d'apprendre des représentations intéressantes des mots. En cela, les tâches ne sont là que pour forcer le modèle à

4. La complexité (en temps de calcul) est logarithmique par rapport à la taille du vocabulaire.
5. Les meilleurs résultats sont obtenus avec $k = 4$.

apprendre les caractéristiques des mots via leur distribution dans le langage. En effet, les plongements de mots réussissent à capturer des relations complexes entre les mots, notamment syntaxiques et sémantiques [Mikolov *et al.*, 2013b]. Une fois les plongements de mots appris, il est possible d'observer les relations apprises en effectuant des opérations simples. Par exemple, ils montrent que l'on a $vecteur(\text{“}Roi\text{”}) - vecteur(\text{“}Homme\text{”}) + vecteur(\text{“}Femme\text{”}) \approx vecteur(\text{“}Reine\text{”})$. Les plongements ainsi obtenus sont comparés aux autres systèmes existants sur leur faculté à capturer les relations entre les mots, en d'autres termes à répondre à des questions de la forme « quel mot est à Z ce que Y est à X », soit pour reprendre l'exemple précédent dont le résultat était *Reine* : « quel mot est à *Femme* ce que *Roi* est à *Homme* » ?

Ces travaux ont donné lieu à la publication de l'implémentation de ce modèle, appelé word2vec [6], ainsi que des vecteurs de mots ainsi entraînés.

De manière similaire, [Pennington *et al.*, 2014] proposent GloVe [7], un outil ainsi que des vecteurs de mots entraînés, tirant parti à la fois des approches statistiques basées sur des matrices de co-occurrences, et des approches se servant du contexte proche, comme word2vec.

Ces travaux ont fourni une méthode et des outils pour entraîner, et évaluer les plongements de mots. De plus, ils ont popularisé l'idée de séparer l'entraînement des plongements de mots d'une part, et des tâches subséquentes de traitement du langage d'autre part. Cette approche en deux étapes, d'abord un pré-entraînement puis l'apprentissage de la tâche elle-même a permis des gains en performances sur une grande variété de tâche, à un coût en temps de calcul moindre, grâce aux représentations pré-entraînées.

2.2.2 Modèles de langage

Un modèle de langage est un modèle probabiliste appliqué aux séquences de mots. Soit le vocabulaire $\mathcal{V}$, à toute séquence de mots $\{w_1, \ldots, w_m\} \in \mathcal{V}^m$ de longueur m quelconque, est associée la probabilité $P(w_1, \ldots, w_m)$.

Comme pour les plongements de mots (sect. 2.2.1), l'apprentissage de tels modèles repose sur *l'hypothèse distributionnelle* [Firth, 1957] : le contexte d'un mot nous renseigne sur ce mot.

Les modèles dits n-gramme prennent comme contexte les $n-1$ mots précédents. Un modèle 3-gramme (ou trigramme) calcule donc la probabilité de chaque mot sachant les

6. https://code.google.com/archive/p/word2vec/
7. https://nlp.stanford.edu/projects/glove/

2 mots précédents *i.e.* $P_{3gram}(w_i \mid w_{i-1}, w_{i-2})$ où w_i est le $i^{\text{ième}}$ mot de la séquence. La probabilité de la séquence complète est alors le produit des probabilités de chaque mot, *i.e.* :

$$\begin{aligned} P_{\text{3-gram}}(w_1, \ldots, w_m) = P(w_1) &\times P\left(w_2 \mid w_1\right) \\ &\times P\left(w_3 \mid w_2, w_1\right)) \\ &\times \ldots \\ &\times P\left(w_m \mid w_{m-1}, w_{m-2}\right) \end{aligned}$$

Ou plus généralement, pour un modèle n-gramme :

$$P_{n\text{-gram}}(w_1, \ldots, w_m) = \prod_{i=1}^{m} P\left(w_i \mid w_{i-1}, \ldots, w_{i-(n-1)}\right)$$

Cependant, les modèles n-gramme, reposant sur les statistiques d'occurrences des mots ne peuvent modéliser que les probabilités des n-gramme étant effectivement présents dans les données. Une solution à ce problème est proposé par les modèles dits *back-off* n-gramme [Katz, 1987] et consiste à répartir la masse de probabilité des n-grammes partageant le même contexte. Par exemple, un trigramme qui n'aurait pas été observé dans les données est associé à une probabilité non nulle du moment qu'un sous-ensemble de ce trigramme (donc un bigramme ou un unigramme) a été observé. Ainsi, le modèle ne donne des probabilités nulles que lorsque aucun des n mots du n-gramme n'a été observé – ce qui est moins problématique dans un cas aussi improbable.

[Bengio *et al.*, 2000] proposent une approche neuronale de modèle de langage basée sur un réseau à propagation avant. Leur modèle utilise un contexte de taille fixe, c'est-à-dire que la distribution de probabilité est exprimée en fonction des n mots précédents, à la manière d'un modèle n-gramme. Ils obtiennent de meilleures performances que les modèles n-gramme, à taille de contexte n égale. De plus, alors que les modèles n-gramme atteignent leurs meilleures performances pour $n = 3$, leur modèle neuronal profite d'un contexte plus grand, et s'améliore jusqu'à un contexte de $n = 5$.

Ce travail a été précurseur dans sa manière d'approcher l'apprentissage des plongements de mots à partir d'un modèle de langage. Leurs contributions s'articulent autour de 3 axes :

1. **Associer chaque mot** w_t de la séquence $w_1, \ldots, w_N$ $(\forall i, w_i \in \mathcal{V})$ à une ***représen-***

tation vectorielle $\boldsymbol{x}_t \in \mathbb{R}^{d_{emb}}$.

$$\begin{aligned} \boldsymbol{x}_t &= onehot(w_t)^\top \mathbf{W}_{emb} \\ \mathbf{E} &\in \mathbb{R}^{|\mathcal{V}| \times d_{emb}} \end{aligned} \tag{2.10}$$

où $\mathbf{E}$ est la matrice de poids des plongements de mots.

2. **Exprimer la *loi de probabilité*** du mot courant w_t en **fonction des représentations des** $n-1$ **mots précédents** :

$$\begin{aligned} p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) &= g\left(f\left(\tilde{\boldsymbol{x}}_t\right)\right) \\ \tilde{\boldsymbol{x}}_t &= (\boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_{t-n+1}) \end{aligned} \tag{2.11}$$

f est une couche *tangente hyperbolique* de taille h avec *connexion directe*[8] :

$$\begin{aligned} \hat{\boldsymbol{y}} &= f(\tilde{\boldsymbol{x}}_t) = \boldsymbol{b} + \mathbf{W}\tilde{\boldsymbol{x}}_t + \mathbf{V}\tanh\left(\mathbf{U}\tilde{\boldsymbol{x}}_t + \boldsymbol{d}\right) \\ \mathbf{W} &\in \mathbb{R}^{|\mathcal{V}| \times (n-1)d_{emb}} \\ \mathbf{V} &\in \mathbb{R}^{|\mathcal{V}| \times h} \\ \mathbf{U} &\in \mathbb{R}^{h \times (n-1)d_{emb}} \end{aligned} \tag{2.12}$$

La fonction g convertit la sortie de f, un vecteur de dimension $|\mathcal{V}|$, en une distribution de probabilité grâce à la fonction *softmax*[9] :

$$\begin{aligned} p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) &= g(\hat{\boldsymbol{y}}) = \mathrm{softmax}(\hat{\boldsymbol{y}}) \\ \mathrm{softmax}(\hat{\boldsymbol{y}}) &= \frac{\exp \hat{\boldsymbol{y}}}{\sum_j \exp \hat{\boldsymbol{y}}_j} \end{aligned} \tag{2.13}$$

3. **Apprendre conjointement la *représentation* et la *loi de probabilité***. Le modèle entier – c'est-à-dire les paramètres $\theta = (\mathbf{E}, \mathbf{U}, \mathbf{V}, \mathbf{W}, \boldsymbol{b}, \boldsymbol{d})$ – est entraîné à maximiser la vraisemblance des données. On exprime la vraisemblance séquence $w_1, \ldots, w_N$, de manière *auto-régressive* :

$$\max_\theta p(w_1, \ldots, w_N) = \frac{1}{N} \prod_{t=1}^{N} p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) \tag{2.14}$$

8. [Bengio *et al.*, 2000] appellent *connexion directe* la connexion entre la sortie d'une fonction d'activation *i.e.* $\tanh(\mathbf{U}\boldsymbol{x})$ et une projection de son entrée *i.e.* $\mathbf{W}\boldsymbol{x}$. Dans leurs expériences, les connexions directes permettent un apprentissage plus rapide, au prix moins bonne généralisation.

9. La fonction retourne un vecteur $\boldsymbol{v}$ tel que $\forall i, \boldsymbol{v}_i \in [0,1]$ et $\sum_i \boldsymbol{v}_i \equiv 1$ ce qui permet de modéliser une distribution de probabilité.

Ce qui est équivalent à minimiser la *log vraisemblance négative*[10]

(*negative log likelihood*). Nous définissons donc la fonction de coût $\mathcal{L}(\theta)$:

$$\min_{\theta} \mathcal{L}(\theta) = -\frac{1}{N} \sum_{t=1}^{N} \log p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) \tag{2.15}$$

Les modèles de langages neuronaux ont ensuite été appliqués, par exemple au *rescoring* ASR par [Schwenk & Gauvain, 2005]. Ceci consiste à calculer la probabilité d'une séquence (donc en quelque sorte, un score) d'après un modèle de langage. Dans le contexte de la reconnaissance automatique de la parole (ASR) le modèle de langage est utilisé pour comparer plusieurs séquences considérées probables par le modèle acoustique. Ceci permet, en plus de la prédiction acoustique, d'ajouter des composantes syntaxiques et sémantiques portées par le modèle de langage.

Afin de se soustraire de la contrainte d'un contexte de taille fixe, [Mikolov *et al.*, 2010] proposent un modèle de langage basé sur un RNN, permettant de traiter des séquences de longueur variable. Les contributions de [Mikolov *et al.*, 2010] par rapport à [Bengio *et al.*, 2000] concernent : (i) l'expression des probabilités en fonction de tous les mots précédents, (eq. 2.11 remplacée par 2.16) ; (ii) la couche f, remplacée par un RNN mono-couche avec état $\boldsymbol{h}_t^e$ (eq. 2.12 devient 2.18).

$$p(w_t \mid w_{t-1}, \ldots, w_1) = g\left(f\left(\tilde{\boldsymbol{x}}_t\right)\right) \tag{2.16}$$

$$f(\tilde{\boldsymbol{x}}_t) = \mathbf{V}\boldsymbol{h}_t^e \tag{2.17}$$

$$\boldsymbol{h}_t^e = \sigma(\mathbf{V}\boldsymbol{x} + \mathbf{W}\boldsymbol{h}_{t-1}^e) \tag{2.18}$$

$$\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}) \tag{2.19}$$

L'évolution et le succès des modèles de langage ont été précurseurs au développement des modèles de traduction et de résumé automatique. En effet, ceux-ci sont aux croisements de l'apprentissage des représentations denses des mots et des séquences, et de la prédiction de mots, ouvrant la voie aux modèles de génération de texte.

10. Les *log probabilité i.e.* $\log p$ sont préférées aux probabilités p pour exprimer la fonction de coût $\mathcal{L}$ afin (i) de simplifier les expressions ; de réduire le temps de calcul : le produit devient une somme ce qui plus efficace ; et (iii) améliorer la stabilité numérique en contournant la difficulté à représenter les probabilités proches de zéro, qui deviennent alors de grandes valeurs négatives, plus aisément représentées.

2.2.3 Représentation de séquences

Une des problématiques des approches neuronales pour le traitement automatique du langage naturel est de traiter des entrées de taille variable. C'est par exemple le cas lorsque, dans le contexte de résumé ou de traduction automatique, toutes les phrases du jeu de données n'ont pas la même longueur en nombre de mots. Cependant, les architectures de réseaux de neurones doivent être définis avec des matrices de dimension fixe : il n'est pas question de changer le nombre de paramètres du modèle d'une paire d'entraînement à l'autre. Par conséquent, il est pertinent de chercher à calculer des représentations de taille fixe à partir de séquences de taille variable.

Une première idée consiste alors à partager les paramètres de la couche d'entrée entre les mots de la séquence. Nous calculons ainsi, pour chaque mot, sa représentation vectorielle, c'est-à-dire son plongement tel que discuté en section 2.2.1. Contrairement aux réseaux de neurones sans partage de paramètres abordés en section 2.1.3 (*e.g.* fig.2.3) où chaque entrée était associée à sa propre matrice de paramètres, on utilise ici la même matrice pour chacun des mots. Ceci permet de mettre en commun l'apprentissage des plongements de mots. En d'autres termes, la couche d'entrée calcule les plongements de mots indépendamment de leur position dans la séquence.

Une fois les plongements calculés, nous obtenons une séquence de vecteurs, mais toujours de taille variable.

Une fonction qui calcule une représentation $\boldsymbol{h} \in \mathbb{R}^{d_{model}}$ de dimension d_{model} fixe à partir d'une séquence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ de taille variable est appelée un *encodeur* (eq. 2.20), et on a $\boldsymbol{h} = \text{encodeur}(\boldsymbol{x})$. Une simple somme pourrait alors être considérée comme un *encodeur* *e.g.* eq. 2.21. Dans les paragraphes suivants, nous présentons les approches communément utilisées.

Réseaux de neurones récurrents (RNN) Introduits en section 2.1.4, les RNN peuvent jouer le rôle d'encodeur. En effet, un RNN parcourt une séquence d'entrée et maintient un état interne h_i à chaque pas i. Plusieurs approches sont alors possibles pour en tirer une représentation de la séquence. La première – et aussi la plus simple – consiste à représenter la séquence grâce au dernier état du RNN *i.e.* h_n (eq. 2.22). Le RNN ayant analyser la séquence dans son intégralité au pas n, on peut penser que l'état correspondant représente la séquence entière. Cependant, une telle représentation peut accorder trop d'importance aux derniers mots (proches de n) par rapport aux premiers. Pour contrer ce problème, il est possible d'agréger les états du *RNN* en calculant, pour chaque dimension, la moyenne

(eq. 2.23) ou le maximum (eq. 2.24).

$$\text{encodeur} : \boldsymbol{\mathcal{P}}(\mathbb{R}^{d_{emb}}) \mapsto \mathbb{R}^{d_{model}} \tag{2.20}$$

$$\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d_{emb}}$$

$$\text{encodeur}_{sum}(\boldsymbol{x}) = \sum \boldsymbol{x}_i \tag{2.21}$$

$$\text{encodeur}_{\text{RNN-last}}(\boldsymbol{x}) = \boldsymbol{h}_n \tag{2.22}$$

$$\text{avec } \boldsymbol{h}_i = RNN(\boldsymbol{x}_i, \boldsymbol{h}_{i-1})$$

$$\text{encodeur}_{\text{RNN-mean}}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{h}_i \tag{2.23}$$

$$\text{encodeur}_{\text{RNN-max}}(\boldsymbol{x}) = [m_1, \ldots, m_{d_{model}}]^\top \tag{2.24}$$

$$\text{avec } m_d = \max_{i \in [1,n]} \boldsymbol{h}_{i,d}$$

où $\boldsymbol{h}_{i,d}$ désigne la dimension d de l'état au pas i

RNN Bidirectionnels (*Bi-RNN*) [Bahdanau *et al.*, 2014] proposent d'utiliser un RNN bidirectionnel, c'est-à-dire d'ajouter un second RNN à l'encodeur qui traite la séquence dans le sens inverse c'est-à-dire de x_n à x_1. L'encodeur est composé de deux RNN, l'un à l'endroit ($\overrightarrow{f^e}$, eq. 2.25) et l'autre à l'envers ($\overleftarrow{f^e}$, eq. 2.25). L'état de l'encodeur est alors la concaténation des états de chacun des RNN (eq. 2.27).

$$\overrightarrow{\boldsymbol{h}_i^e} = \overrightarrow{f^e}\left(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}^e\right), \ \boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \tag{2.25}$$

$$\overleftarrow{\boldsymbol{h}_i^e} = \overleftarrow{f^e}\left(\overleftarrow{\boldsymbol{x}_i}, \overleftarrow{\boldsymbol{h}_{i-1}^e}\right), \ \overleftarrow{\boldsymbol{x}} = (\boldsymbol{x}_n, \ldots, \boldsymbol{x}_1) \tag{2.26}$$

$$\boldsymbol{h}_i^e = \left[\overrightarrow{\boldsymbol{h}_i^e}^\top ; \overleftarrow{\boldsymbol{h}_i^e}^\top\right]^\top \tag{2.27}$$

Représentation universelle de phrases De la même manière que l'on a cherché à entraîner des plongements de mots universels, c'est-à-dire pouvant servir de représentation de mots pour n'importe quelle tâche de traitement de langage, [Kiros *et al.*, 2015] et [Conneau *et al.*, 2017] ont cherché à apprendre des représentations universelles de phrases.

[Kiros *et al.*, 2015] entraînent des vecteurs de phrases de manière similaires aux entraînements de plongements de mots, c'est-à-dire en essayant de prédire une phrase à

partir de son contexte de manière non supervisée. Les représentations de phrases ainsi obtenues sont évaluées sur leurs capacités à mesurer les similarités sémantiques (entre autres).

À l'inverse, [Conneau *et al.*, 2017] optent pour une approche supervisée basée sur la tâche *d'inférence du langage naturel* (ou *NLI* pour *Natural Language Inference*) qui consiste à déterminer si deux séquences, une *prémisse* et une *hypothèse* sont en accord, en contradiction ou neutres. Leurs modèles utilisent alors des encodeurs pour obtenir une représentation dense de chaque phrase. La couche de sortie effectue une classification en trois catégories (accord, contradiction et neutre) à partir de ces représentations. Les encodeurs ainsi entraînés servent ensuite de base pour l'entraînement de 12 tâches de transfert afin de montrer les capacités de généralisation à partir des représentations apprises.

Transformer Encodeur Le Transformer est un modèle présenté par [Vaswani *et al.*, 2017] en réponse aux limitations des RNN. Ce modèle – que nous présentons en détails en section 2.4 – a notamment été utilisé afin d'obtenir des représentations de séquences, par exemple dans le cadre de Bert [Devlin *et al.*, 2019], abordé dans la section 2.5.

2.3 Le modèle *encodeur-décodeur* (= seq2seq)

2.3.1 Modèle de base

Nous avons vu, d'une part, les *encodeurs* (sec. 2.2.3) qui permettent d'obtenir une représentation vectorielle une séquence de mots. D'autre part, nous avons présenté les *modèles de langage* (sec. 2.2.2) qui modélisent la probabilité d'apparition d'un mot dans un contexte, et permettent donc de prédire le mot dans une phrase.

Dans ce contexte [Cho *et al.*, 2014] et [Sutskever *et al.*, 2014] ont proposé une architecture neuronale permettant de passer d'une séquence d'entrée à une séquence de sortie de taille éventuellement différente. Ces modèles dits *sequence-to-sequence* (ou *seq2seq*) sont la base des approches neuronales pour la traduction et le résumé automatique.

Ces modèles sont composés de deux parties : un *encodeur* chargé de calculer une représentation de la séquence d'entrée, et un *décodeur*, qui, à partir de l'entrée *encodée* cherche à prédire la séquence cible, à la manière d'un modèle de langage. Pour cette raison, ces modèles sont également couramment appelés *encodeur-décodeur*.

Les encodeurs de [Cho *et al.*, 2014] et [Sutskever *et al.*, 2014] utilisent des RNN.

Toutefois, comme indiqué précédemment, à partir du constat de [Bengio *et al.*, 1994] quant à la difficulté des RNN à apprendre des relations à longue distance, [Sutskever *et al.*, 2014] utilise une variante du RNN classique appelée Long Short Term Memory (LSTM). [Cho *et al.*, 2014] proposent également une variante : Gated Reccurent Unit (GRU).

L'encodeur calcule une représentation en dimension fixe appelée le *vecteur contexte* $\boldsymbol{c}^e \in \mathbb{R}^{d_{enc}}$, à partir de l'entrée $\boldsymbol{x} = (x_1, \ldots, x_n)$ du dernier état de l'encodeur $\boldsymbol{h}^e_n$ (*cf.* encodeur$_{last}$ eq. 2.22).

$$\begin{aligned} \boldsymbol{c}^e &= \tanh(\mathbf{V}^e \boldsymbol{h}^e_n) \\ \text{avec } \boldsymbol{h}^e_i &= f^e\left(\boldsymbol{h}^e_{i-1}, \boldsymbol{x}_i\right) \\ \boldsymbol{h}^e_0 &= \{0\}^H \end{aligned}$$

où f^e est un RNN (de type GRU ou LSTM) qui itère sur les vecteurs de mots $\boldsymbol{x}_i \in \mathbb{R}^{d_{emb}}$

Le vecteur contexte de l'encodeur, $\boldsymbol{c}^e$ est ensuite transmis au décodeur qui calcule la distribution de sortie mot à mot :

$$\begin{aligned} p(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1, \boldsymbol{x}) &= g(\boldsymbol{h}^d_t, \boldsymbol{c}^e) \\ \text{avec } \boldsymbol{h}^d_t &= f^d(\boldsymbol{h}^d_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}^e) \\ \boldsymbol{h}^d_0 &= \tanh(\mathbf{V}^d \boldsymbol{c}^e) \end{aligned}$$

La fonction décodeur f^d est par exemple un RNN (GRU ou LSTM). La fonction g, que l'on appelle le *générateur* (eq. 2.28) produit la distribution de probabilité grâce à une couche *softmax* permettant de produire une distribution de probabilité à partir de valeurs quelconques (eq. 2.29). En d'autre terme, le *générateur* associe chaque mot à une probabilité à partir de la couche de sortie du décodeur.

$$g(\boldsymbol{h}^d_t, \boldsymbol{c}^e) = \text{softmax}(\mathbf{W}\boldsymbol{h}^d_t + \mathbf{V}\boldsymbol{c}^e + b) \tag{2.28}$$

$$\text{softmax}(\boldsymbol{x}) = \frac{\exp \boldsymbol{x}}{\sum_j \exp \boldsymbol{x}_j} \tag{2.29}$$

La première entrée du décodeur est un mot spécial désignant le début de la séquence (noté `BOS` ou *begining of sequence*). De même, le dernier mot à prédire est un mot spécial de fin de séquence (`EOS`, pour *end of sequence*) qui permet au modèle de terminer la

génération.

Le modèle complet, c'est-à-dire l'encoder et le décodeur sont entraînés conjointement à minimiser la *log-vraisemblance négative* de la séquence cible $\boldsymbol{y}$, à la manière d'un modèle de langage (*cf.* sec. 2.2.2, eq. 2.15) *i.e.* :

$$\min_{\theta} \boldsymbol{\mathcal{L}}(\theta) = -\frac{1}{|\boldsymbol{y}|} \sum_{t=1}^{|\boldsymbol{y}|} \log p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \tag{2.30}$$

2.3.2 Le mécanisme d'attention

[Bahdanau *et al.*, 2014] suggèrent que le fait d'utiliser le même *vecteur contexte* $\boldsymbol{c}^e$ à chaque pas au sein du décodeur, comme dans [Cho *et al.*, 2014], pourrait constituer une limite aux modèles *encodeur-décodeur*. Ils introduisent alors le mécanisme d'attention qui permet de calculer un vecteur contexte $\boldsymbol{c}_t^e$ à chaque instant t comme étant la somme des états $\boldsymbol{h}_i^e$ de l'encodeur pondérés par l'attention α_{ti} (*cf.* eq. 2.32). En d'autres termes, à un instant t, le vecteur $\alpha_t \in \mathbb{R}^n$ est la distribution de l'attention portée à chaque entrée. Par exemple, pour une séquence d'entrée de trois mots *i.e.* $n = 3, \boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, l'encodeur calcule les états $\boldsymbol{h}_1^e, \boldsymbol{h}_2^e, \boldsymbol{h}_3^e$, à un instant t quelconque la distribution de l'attention peut être : $\alpha_t = [1, 0, 0]^\top$ ce qui reviendrait à considérer $\boldsymbol{c}_t^d = \boldsymbol{h}_1^e$ ou alors $\alpha_t = [1/3, 1/3, 1/3]^\top$ ce qui reviendrait à utiliser un encodeur par moyenne des états (voir 2.23).

La pondération de l'attention est calculée au sein d'une couche cachée à partir l'état du décodeur à l'instant t (*i.e.* $\boldsymbol{h}_t^d$), et les états de l'encodeur (*i.e.* $\boldsymbol{h}_i^e \; \forall i \in [\![1, n]\!]$). L'attention est d'abord calculée sous forme de valeur *d'énergie* e_{ti} (eq. 2.34) puis de distribution α_t grâce à une couche softmax (eq. 2.33) *i.e.* $\alpha_{ti} \in [0, 1], \sum_i \alpha_{ti} = 1$. On peut voir, en figure 2.7, la distribution de l'attention, dans le cadre de traduction de l'anglais vers le français. On observe alors que ce mécanisme parvient à identifier les mots importants de la source, y compris lorsque l'ordre est renversé comme dans le cas des adjectifs et des noms : le sens de la diagonale s'inverse pour *« European Economic Area »* qui devient *« zone économique européenne »*.

$$p(y_t \mid y_{t-1}, \ldots, y_1, \boldsymbol{x}) = g(\boldsymbol{h}_t^d, \boldsymbol{c}_t^e) \tag{2.31}$$

$$\boldsymbol{c}_t^e = \sum_{i=1}^{n} \alpha_{ti} \boldsymbol{h}_i^e \tag{2.32}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{n} \exp(e_{tj})} \tag{2.33}$$

$$e_{ti} = v_a^\top \tanh(\mathbf{W}_{attn}^e \boldsymbol{h}_t^d + \mathbf{V}_{attn}^e \boldsymbol{h}_i^e) \tag{2.34}$$

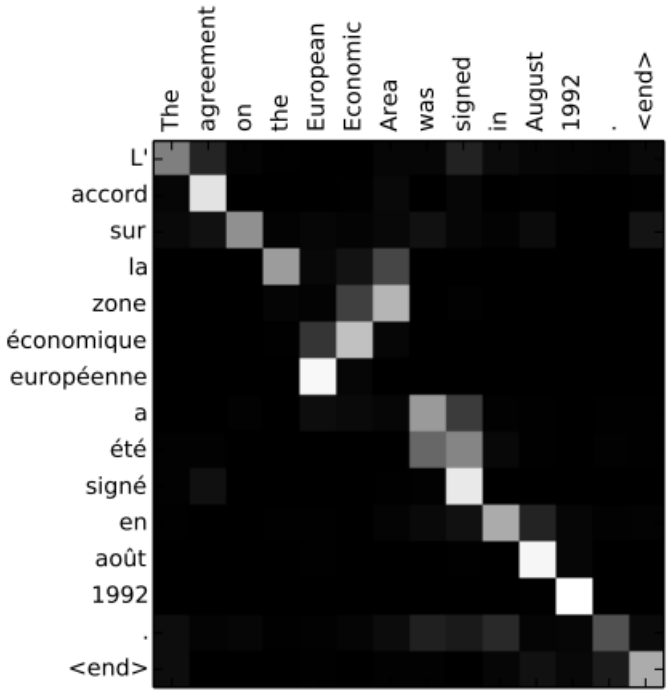


FIGURE 2.7 – Matrice représentant la distribution de l'attention *alpha* dans le contexte de traduction d'une phrase de l'anglais vers le français. Chaque pixel représente l'attention $\alpha_{t,i}$ portée i-ième mot source (en anglais) par rapport au t-ième mot cible (en français)

2.4 Le modèle Transformer

Les modèles de traitement automatique du langage naturel que nous avons vus jusqu'ici ont principalement utilisé les réseaux de neurones récurrents (RNN). De plus, les modèles *encodeur-décodeur* et les modèles de langage basés sur des RNN ont permis des amélio-

rations majeures des performances notamment en transcription automatique, traduction neuronale et en résumé automatique. Le mécanisme d'attention a permis d'améliorer encore le modèle encodeur-décodeur, et de nombreuses variantes ont alors été étudiées.

Toutefois, [VASWANI *et al.*, 2017] suggèrent que les récurrences limitent les performances de tels modèles, car la séquentialité des opérations empêche la parallélisation de celles-ci. Ils suggèrent alors de remplacer les RNN de l'encodeur, et du décodeur par des couches d'*auto-attention*. Les états $\boldsymbol{h}_i^e$ de l'encodeur et $\boldsymbol{h}_t^d$ ne dépendent plus des états précédents (resp. $\boldsymbol{h}_{i-1}^e$, et $\boldsymbol{h}_{t-1}^d$) et peuvent être calculés en parallèle. Ceci permet de baisser le temps d'entraînement des modèles, de distribuer plus simplement les calculs entre plusieurs périphériques ou nœuds de calcul, et enfin de faciliter l'apprentissage des relations entre les mots distants.

[VASWANI *et al.*, 2017] introduisent alors le mécanisme d'attention *multi-têtes* (*multi-head attention*). Cette technique repose sur un ensemble de plusieurs couches en parallèle (les têtes) d'attention (eq. 2.35). Ils proposent les termes *valeurs*, *requêtes* et *clés* comme opérandes de l'opération attention, notés respectivement V, Q et K. Chaque tête d'attention dispose de ses propres poids, utilisés pour projeter chaque opérande (eq. 2.36). L'attention est alors la somme des valeurs V pondérées par la *compatibilité* entre les projections de Q et de K (eq. 2.37).

$$\text{MultiHeadAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{concat}\left(\text{head}_1,\ldots,\text{head}_h\right)\mathbf{W}^O \tag{2.35}$$

$$\text{head}_i = \text{attention}\left(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V\right) \tag{2.36}$$

$$\text{attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.37}$$

Cette définition de l'attention est proche de celle de [BAHDANAU *et al.*, 2014] et de ce que nous avons appelé le *vecteur contexte* ($\boldsymbol{c}_t^e$) en cela qu'il s'agit d'une somme des *annotations* ($\boldsymbol{h}_i^e$, correspondant ici à V), pondérée par la distribution α_t, calculée entre l'état du décodeur $\boldsymbol{h}_t^d$ (ici, Q) et l'état de l'encodeur $\boldsymbol{h}_i^e$ (ici K). L'attention telle que décrite par [BAHDANAU *et al.*, 2014] correspond donc, dans la notation de [VASWANI *et al.*, 2017] à $K = V$ et $h = 1$ (où h.est ici le nombre de têtes d'attention).

La définition de [VASWANI *et al.*, 2017] permet également d'introduire la notion *d'auto-attention* c'est-à-dire $Q = K$ et qui est utilisée dans les couches basses de l'encodeur et du décodeur (respectivement en rouge et bleu dans la figure 2.8).

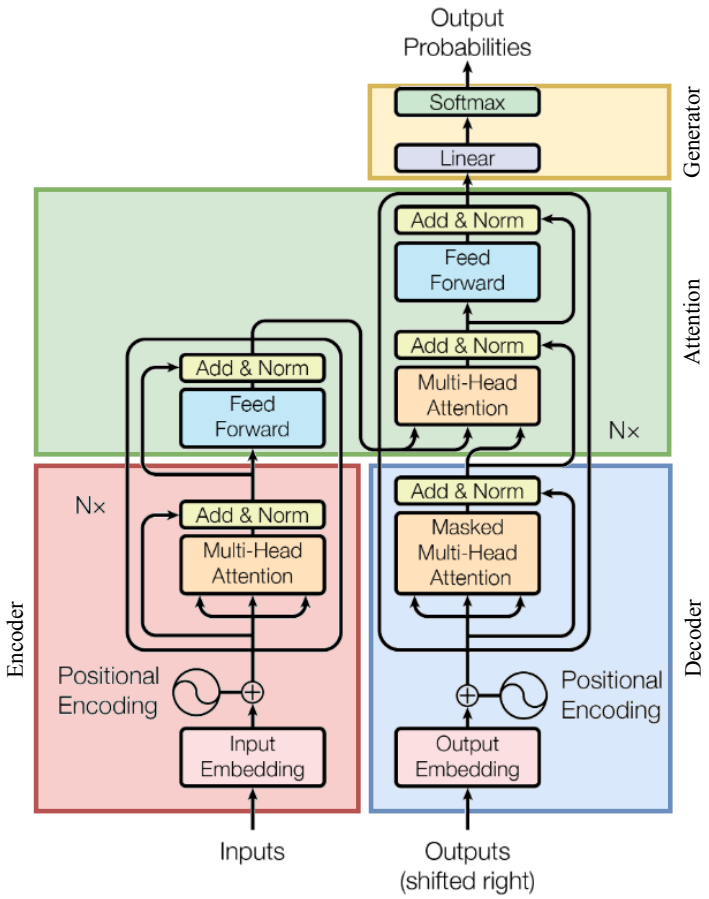


FIGURE 2.8 – Architecture du modèle TRANSFORMER de [VASWANI *et al.*, 2017]. Les RNN de *l'encodeur* et du *décodeur* sont tous deux remplacés par des couches d'auto-attention *multi-têtes* (resp. en rouge et en bleu). En vert, l'attention est calculée entre l'encodeur et le décodeur afin de produire, dans le *générateur* (en jaune) la distribution de sortie

2.5 L'apprentissage auto-supervisé du langage

Nous avons vu dans les sections 2.3 et 2.4 deux approches permettant d'entraîner des modèles *sequence-to-sequence*, l'une basée sur les RNN, l'autre sur *l'auto-attention*. Si ces modèles se distinguent par leur architecture, leurs entraînements demeurent similaires : il s'agit d'apprentissage supervisé, c'est-à-dire que l'on entraîne ces modèles à maximiser la vraisemblance de la séquence *cible* à partir d'une séquence *source*. De fait, ceci nécessite de constituer des jeux de données avec des paires d'entraînements $(source, cible)$ alignées. Par exemple, dans le cadre de traduction automatique, on cherchera des paires de phrases en français et en anglais telle que l'une soit la traduction de l'autre. Dans le contexte du résumé automatique, cela nécessite de disposer de textes d'une part ainsi que de leurs résumés. On voit alors aisément que ceci constitue une limitation importante. En particulier, on constate une disproportion dans les quantités et la facilité d'accès des données alignées par rapport aux données non alignées. Par exemple, nous pourrions obtenir en parcourant Wikipedia un très grand nombre de phrases en anglais et en français. Cependant, les articles n'étant pas des traductions littérales, il serait difficile d'en extraire des paires d'entraînement pour de l'apprentissage supervisé.

L'apprentissage *auto-supervisé* consiste à entraîner des modèles non pas à partir de paires $(source, cible)$ mais simplement de séquences non alignées. Ceci permet de se libérer des contraintes spécifiques aux tâches supervisées qui pèsent sur la généralisation des modèles et l'abondance des données d'entraînement. On distingue alors deux approches d'apprentissage *auto-supervisé* :

— **Les modèles de langage Auto-régressif (AR).** Comme abordé en section 2.2.2, un modèle de langage est entraîné à prédire un mot de la séquence à partir des mots précédents. En d'autre termes, pour une séquence $\boldsymbol{x} = [x_1, \ldots, x_n]$, on cherche à maximiser probabilité de $\boldsymbol{x}$ *i.e.* $\max_\theta \log p_\theta(\boldsymbol{x}) = \sum_{i=1}^{N} \log p_\theta(x_i \mid \boldsymbol{x}_{<i})$. Il s'agit de l'approche utilisée par (Bengio *et al.*, 2000, *cf.* sec. 2.2.2) pour apprendre les représentations de mots (*word embeddings*) grâce à l'entraînement d'un modèle de langage et plus récemment pour les pré-entraînements de GPT [Radford *et al.*, 2018], GPT-2 [Radford *et al.*, 2019] et GPT-3 [Brown *et al.*, 2020].

— **Auto-encodeur (AE).** L'objectif d'un *auto-encodeur* consiste à reconstituer la séquence $\boldsymbol{x}$ à partir d'une version *bruitée* $\tilde{\boldsymbol{x}} = \tilde{f}(\boldsymbol{x})$ de celle-ci. On maximise alors la probabilité de $\boldsymbol{x}$ sachant $\tilde{\boldsymbol{x}}$: $\max_\theta \log p_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \sum_{i=1}^{N} \log p_\theta(x_i \mid \tilde{\boldsymbol{x}})$. Pour *bruiter* la source, on peut par exemple remplacer aléatoirement des mots de la séquence

par un mot spécial `MASK` comme c'est le cas dans BERT [DEVLIN *et al.*, 2019] ou permuter les phrases de la séquence comme dans BART [LEWIS *et al.*, 2020].

Ces deux approches ont des objectifs, et des contextes d'applications bien différents. D'une part, les modèles AR ne prennent en compte que le *contexte gauche* c'est-à-dire les mots précédents. Dans le cadre de génération de texte, cela fait parfaitement sens : les mots sont générés les uns à la suite des autres, il n'est pas question de prendre en compte les mots suivants. Quant à eux, les modèles AE utilisent l'intégralité du contexte (gauche et droit) ce qui ne limite pas l'apprentissage des relations entre les mots à leur ordre d'apparition. On a donc une approche (AR) focalisée sur la génération et dont la représentation est limitée, et une autre (AE) concentrée sur la représentation mais peu adaptée à la génération. Plusieurs modèles ont été proposés pour généraliser ces deux approches, et ainsi tirer le meilleur des deux mondes :

— **Débruitage avec encodeur-décodeur :** Le modèle BART de [LEWIS *et al.*, 2020] est un TRANSFORMER complet, c'est-à-dire avec encodeur et décodeur. La source bruitée est passée à l'encodeur et le décodeur doit prédire la séquence originale. En cela, le modèle se comporte globalement à la manière d'un auto-encodeur tout en comportant un décodeur auto-régressif. Il peut ainsi facilement être pré-entraîné sur une tâche de débruitage, puis être spécialisé sur une tâche plus spécifique telle que le résumé automatique, sans modification de l'architecture.

— **Apprentissage auto-régressif bidirectionnel :** [YANG *et al.*, 2019] présentent XLNET, un modèle auto-régressif entraîné à partir de permutations des séquences. En effet, afin de se libérer de la contrainte du *contexte gauche* des modèles auto-régressifs, ils proposent de présenter au modèle les séquences d'entraînement dans chacune de leurs permutations. Ceci permet de généraliser l'apprentissage à n'importe quel ordre de séquence, donc en particulier d'obtenir un modèle à la fois auto-régressif et bidirectionnel.

— **Modèle de langage Universel :** [DONG *et al.*, 2019] proposent avec UNILM d'unifier trois tâches de modélisation du langage : unidirectionnel (*e.g.* les modèles AR) ; bidirectionnel (*e.g.* les AE) ; et *sequence-to-sequence*. Ils combinent alors les jeux de données correspondants aux trois tâches et mutualisent l'apprentissage. Pour chaque tâche, des *masques d'attention* permettent de restreindre l'accès du modèle au contexte correspondant. Par exemple, les mots futurs sont masqués pour la tâche unidirectionnelle et un masque aléatoire est appliqué pour la tâche bidirectionnelle. Pour la tâche *sequence-to-sequence*, la séquence est composée d'une partie corres-

pondant à la source et d'une autre correspondant à la cible, séparées par un mot spécial `SEP`. La partie source n'est pas masquée, mais le mot à prédire et les suivants le sont (sans quoi la tâche est triviale).

L'apprentissage auto-supervisé est souvent utilisé en tant que *pré-entraînement*, avant une phase de spécialisation, appelée le *fine-tuning*. On utilise alors les approches auto-supervisées dans un premier temps afin d'apprendre des représentations générales du langage sur de grands ensembles de données. Ensuite, on réutilise ces représentations dans le cadre de l'apprentissage de tâches plus spécifiques comme le résumé, la génération de question-réponse ou la classification.

Le succès des modèles auto-supervisés présentés dans cette section tient beaucoup à leurs capacités à obtenir des performances proches de l'état de l'art sur différentes tâches de traitement du langage après une courte phase de spécialisation (*few shot*), voire sans spécialisation (*zero shot*).

Le *fine-tuning* pour le résumé abstractif de modèles pré-entraînés de manière auto-supervisée est abordé en section 3.5. De plus, nous présentons l'utilisation de BART dans le cadre de pré-entraînements sur les données UBIQUS en section 5.2. Ce pré-entraînement nous permet de tirer profit de la grande quantité de données non alignées (supérieure de deux ordres de grandeurs).

2.6 Conclusion

Dans ce chapitre, nous avons introduit les notions de réseaux de neurones artificiels et d'apprentissage automatique. Nous nous sommes ensuite intéressé à l'usage des réseaux de neurones dans le cadre de la représentation du langage, c'est-à-dire la modélisation avec des réseaux de neurones de structures du langage comme les mots et les séquences de mots. Ensuite, nous avons vu comment ces représentations sont utilisées pour apprendre des structures plus complexes comme la génération d'une séquence à partir d'une autre, dans le cadre par exemple de traduction automatique ou de résumé avec le modèle *encodeur-décodeur*. Le modèle Transformer présenté ensuite est une variante de l'*encodeur-décodeur* ne s'appuyant plus sur des réseaux de neurones récurrents mais uniquement sur des Mécanismes d'Attention. Dans la dernière section, nous nous sommes intéressé à l'apprentissage auto-supervisé c'est-à-dire sans données alignées.

Chapitre 3

Approches Neuronales du résumé automatique

Après avoir introduit les approches neuronales pour le Traitement Automatique du Langage Naturel au chapitre 2, nous en présentons dans ce chapitre les applications dans le domaine du résumé automatique de texte.

Table des matières

3.1 Modèle *encodeur-décodeur* appliqué au résumé

Le premier modèle *encodeur-décodeur* pour le résumé abstractif a été présenté par [Rush *et al.*, 2015]. Leur modèle – Abs – utilise le mécanisme d'attention de [Bahdanau *et al.*, 2014] à la différence que l'encodeur et le décodeur n'utilisent pas de RNN. L'encodeur représente chaque entrée comme la somme lissée des plongements des $2Q+1$ mots voisins (*cf.* eq. 3.6). Le décodeur quant à lui est un modèle de langage neuronal non récurent (*NNLM*) : $p(y_t \mid \boldsymbol{x}, \boldsymbol{y}_c; \theta)$ où $\boldsymbol{y}_c$ est une fenêtre glissante de taille fixe C sur les mots précédents *i.e.* $\boldsymbol{y}_c = (\boldsymbol{y}_{t-C}, \ldots, \boldsymbol{y}_{t-1})$. La fonction g est le *générateur* : une couche *softmax* (*cf.* 2.28). On observe de légères différences dans le calcul de l'attention par rapport au papier original de [Bahdanau *et al.*, 2014]. L'énergie e_{ti} est calculée à partir des plongements de la fenêtre $\boldsymbol{y}_c$ et de l'entrée $\boldsymbol{x}$ au lieu des sorties de l'encodeur $\boldsymbol{h}^e$ et du décodeur $\boldsymbol{h}^d$, et utilise un *produit scalaire pondéré*[1]. La fenêtre glissante utilise deux plongements de mots $\boldsymbol{y}_c$ et $\boldsymbol{y}'_c$ (respectivement dans les équations 3.2 et 3.5) correspondants à des paramètres distincts.

$$p(y_t \mid \boldsymbol{x}, \boldsymbol{y}_c; \theta) = g(\boldsymbol{h}^d_t, \boldsymbol{c}^e_t) \tag{3.1}$$

$$\boldsymbol{h}^d_t = \tanh(\boldsymbol{y}_c) \tag{3.2}$$

$$\boldsymbol{c}^e_t = \sum_{i=1}^{n} \alpha_{ti} \boldsymbol{h}^e_i \tag{3.3}$$

$$\boldsymbol{\alpha_t} = \text{softmax}(\boldsymbol{e}_t) \tag{3.4}$$

$$e_{ti} = \boldsymbol{x}^\top \mathbf{W} \tilde{\boldsymbol{y}}'_c \tag{3.5}$$

$$\boldsymbol{h}^e_i = \sum_{q=i-Q}^{i+Q} \boldsymbol{x}_i / Q \tag{3.6}$$

Les prédictions sont générées grâce à un algorithme de *recherche en faisceau* (*beam search*, voir Rush *et al.*, 2015 section 4) qui explore plusieurs séquences candidates en parallèle (en général entre 5 et 10).

Une extension est aussi présentée – Abs+ – incluant des caractéristiques extractives. Cette extension permet au modèle d'obtenir un score ROUGE légèrement supérieur (+1 R1) et augmente sensiblement le taux de copie du résumé généré (c'est-à-dire le pourcentage de mot du résumé qui sont aussi présent dans la source).

1. C'est-à-dire une fonction $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \mathbf{W} \boldsymbol{y}$. En comparaison, [Bahdanau *et al.*, 2014] utilisent un *Multi-Layer Perceptron i.e.* $f(\boldsymbol{x}, \boldsymbol{y}) = v^\top \tanh(\mathbf{W}\boldsymbol{x} + \mathbf{V}\boldsymbol{y})$ (*cf.*. eq. 2.34)

Ensuite, [Chopra *et al.*, 2016] ont étendu le modèle de [Rush *et al.*, 2015] avec une architecture plus complexe. En particulier, ils utilisent un encodeur convolutif et le décodeur, utilisant un *NNLM* (non récurrent) dans [Rush *et al.*, 2015] est remplacé par un *RNNLM*. Le mécanisme d'attention est utilisé entre les états du décodeur $\boldsymbol{h}_t^d$ et les entrées encodées $\boldsymbol{h}_i^e$.

Ces deux travaux ont été appliqués avec succès au résumé abstractif de phrase.

3.2 Extensions du modèle *encodeur-décodeur*

Les modèles *encodeur-décodeur* pour le résumé abstractif présentés précédemment (sec. 3.1) étaient entraînés à résumer une seule phrase. Le jeu de données `CNN/DailyMail` a ensuite été proposé par [Nallapati *et al.*, 2016]. Il est composé d'articles issus de *CNN* et du *Daily Mail*. La tâche consiste ensuite à résumer un article de quelques dizaines de phrases en quelques phrases.

L'allongement des séquences sources et cibles a mis au jour des défaillances des modèles *encodeur-décodeur*. En particulier, il a été constaté que ceux-ci tendent à générer beaucoup de répétitions lorsque exposés à des longues séquences [Nallapati *et al.*, 2016 ; Paulus *et al.*, 2017 ; See *et al.*, 2017]. De plus, si le mécanisme d'attention permet au modèle de déterminer les passages les plus importants du texte source, il parvient mal à changer son attention, c'est-à-dire à se concentrer sur une autre partie de la source lorsque la plus importante a été traitée. Ceci accentue les répétitions et empêche le modèle de traiter l'intégralité du document source. Enfin, il est difficile de gérer correctement les mots rares. En effet, qu'il s'agisse de noms propres, d'expressions particulières ou de vocabulaire spécialisé, on peut rencontrer dans la source des mots (ou séquences) pas ou peu vus lors de l'entraînement. Il sera alors difficile, voire impossible, pour le modèle d'inclure ces informations – éventuellement importantes – dans le résumé.

Nous présentons dans cette section des extensions au modèle encodeur-décodeur visant à traiter ces problèmes. L'attention temporelle (sec. 3.2.1) traite le problème des répétitions. Le mécanisme de *couverture* (*coverage*, sec. 3.2.2) vise à s'assurer que le résumé généré traite bien l'intégralité de la source. Enfin, nous verrons le mécanisme de copie (sec. 3.2.3) permettant au décodeur de copier des mots issus de la source directement.

3.2.1 Attention Temporelle

L'idée derrière l'attention *temporelle* est de répartir les attentions successives (des différents instants t) sur les différents mots de la source. En d'autres termes, un mot source recevra moins d'attention à l'instant t s'il en a déjà reçu à $t' < t$.

[Sankaran *et al.*, 2016] introduisent un vecteur *historique* de l'attention β_t comme étant la somme des attentions passées. À chaque instant, l'attention calculée est divisée par l'historique pour obtenir l'attention *temporelle* : α'_t.

$$\beta_t = \sum_{k=1}^{t-1} \alpha_k$$
$$\alpha'_t \propto \frac{\alpha_t}{\beta_t}$$

Cette approche est dite *destructive* en cela qu'elle agit directement sur les valeurs d'attentions. L'attention temporelle a été utilisée au sein de modèles de résumé automatique, notamment dans [Nallapati *et al.*, 2016].

3.2.2 Couverture

[Tu *et al.*, 2016] proposent un mécanisme de *couverture* dont le but est de produire une sortie qui traite bien toute l'entrée. En particulier, l'objectif est d'éviter les répétitions et que le modèle ne se focalise trop sur une partie de l'entrée. Pour ce faire, ils introduisent un *vecteur couverture*, $\boldsymbol{c}_t$, dont les éléments $\boldsymbol{c}_{t,i}$ résument l'attention portée à une entrée $\boldsymbol{h}^e_i$ lors des instants précédant t.

$$\boldsymbol{c}_{t,i} = f_{coverage}(\boldsymbol{c}_{t-1,i}, \alpha_{ti}; \boldsymbol{h}^e_i, \boldsymbol{h}^d_{t-1})$$

où $f_{coverage}$ est une fonction non linéaire – [Tu *et al.*, 2016] utilise une cellule GRU [Cho *et al.*, 2014]. Une variante plus simple a été explorée dans [See *et al.*, 2017] où le vecteur de couverture est simplement la somme des vecteurs attentions α_t précédents *i.e.*

$$c^d_{t,i} = \sum_{j=1}^{t-1} \alpha_{ti}$$

Le vecteur de couverture est ensuite utilisé lors du calcul de l'attention :

$$e_{ti} = v_a^\top \tanh(\mathbf{W}^e_{attn}\boldsymbol{h}^d_t + \mathbf{V}^e_{attn}\boldsymbol{h}^e_i + \mathbf{U}_{cov}\boldsymbol{c}_t)$$

Contrairement à *l'attention temporelle* (sec. 3.2.1), cette approche est dite *informative* en cela que le vecteur couverture est maintenant un paramètre du mécanisme d'attention. Le mécanisme d'attention peut maintenant apprendre à utiliser au mieux le vecteur couverture (grâce aux poids $\mathbf{U}_{cov}$) par opposition à l'attention temporelle où l'historique β_t vient explicitement diviser l'attention α_t.

Le mécanisme de couverture a été adapté aux modèles de résumé automatique par [See *et al.*, 2017] en réponse aux limitations de l'attention temporelle.

3.2.3 Mécanisme de copie

Le mécanisme de copie a été présenté dans différents travaux [Sutskever *et al.*, 2011 ; Luong *et al.*, 2015 ; Vinyals *et al.*, 2015 ; Gulcehre *et al.*, 2016] dans le contexte de la traduction machine. L'objectif est de permettre au modèle de copier un mot à partir de la source au lieu de le générer. Ceci est particulièrement pertinent lorsque l'on est confronté à des mots rares, et insensibles à la traduction comme les noms propres. Ce mécanisme, appliqué au résumé automatique, confère au modèle une composante extractive : le modèle peut utiliser des mots, ou même des passages de la sources dans le résumé.

Le mécanisme de copie est notamment intégré au sein d'un modèle de résumé automatique par [Nallapati *et al.*, 2016] et [See *et al.*, 2017]. Une couche est alors dédiée au calcul de la probabilité (p_{copy}) qui représente la probabilité copier un mot de la source, par opposition à la probabilité $p_{gen} = 1 - p_{copy}$ de générer un mot. L'attention α_t est utilisée pour pondérer la probabilité de copier un mot. Dans [Nallapati *et al.*, 2016] ce mécanisme n'est activé qu'en présence de mots en dehors du vocabulaire ou d'entité nommée. À l'inverse, le modèle [See *et al.*, 2017] combine les probabilités obtenues par le générateur ($P_{vocab}(w)$, eq. 3.8) et par le mécanisme de copie ($P_{copy}(w)$, eq. 3.9) au sein d'une même distribution P (eq. 3.10) :

$$p_{gen} = \sigma(\mathbf{W}_e \boldsymbol{c}_t^e + \mathbf{W}_d \boldsymbol{h}_t^d + \mathbf{W}_x x + b) \tag{3.7}$$

$$P_{gen} = f_{gen}(\boldsymbol{h}_t^d, \boldsymbol{c}^e) \tag{3.8}$$

$$P_{copy} = \sum_{i:w_i=w} \alpha_{ti} \tag{3.9}$$

$$P(w) = p_{gen} \times P_{gen}(w) + (1 - p_{gen}) \times P_{copy}(w) \tag{3.10}$$

3.2.4 Présentation du modèle PointerGen de [See *et al.*, 2017]

Dans cette section nous nous concentrons sur le modèle PointerGen de [See *et al.*, 2017] afin de présenter une architecture complète reprenant les méthodes présentées dans le début de ce chapitre. En effet, le modèle PointerGen consiste en un *encodeur-décodeur* utilisant un *BiLSTM* pour l'encodeur et un LSTM pour le décodeur. De plus, il dispose d'un mécanisme de *couverture* (sec. 3.2.2) et de *copie* (sec. 3.2.3).

Ce modèle est à première vue similaire aux travaux de [Nallapati *et al.*, 2016] mais apporte des changements intéressants. D'abord il est suggéré que *l'attention temporelle* réduit les performances à cause du fait qu'elle « déforme le signal ». En effet, dans l'attention temporelle, on divise explicitement l'attention par l'historique au lieu de laisser le modèle apprendre de lui-même à prendre en compte l'historique comme dans le mécanisme de couverture (voir 3.2.2).

Aussi, comme expliqué dans la section 3.2.3, [See *et al.*, 2017] propose de combiner les probabilités issues du générateur et du mécanisme de copie, au lieu de dissocier les deux cas (voir eq. 3.10). C'est ce mécanisme de copie (qui *pointe* des mots à utiliser) combiné à la génération que les auteurs appellent *Pointeur-Générateur* (*Pointer-Generator*).

Le modèle est représenté par la figure 3.1 et détaillé dans le tableau 3.1.

3.3 Le modèle *Transformer* appliqué au résumé

De même que les modèles encodeur-décodeur basés sur des RNN, le modèle Transformer (*cf.* sec. 2.4) a été développé dans le cadre de la traduction neuronale avant d'être rapidement étendu à d'autres tâches de Traitement Automatique du Langage Naturel (TALN). En particulier, l'objectif même du modèle Transformer – *i.e.* éliminer les calculs séquentiels afin de mieux paralléliser les opérations – est pertinent pour le résumé automatique en cela qu'il permet de traiter des séquences d'entrée, et de sortie, plus longues par rapport aux modèles *encodeur-décodeur* classiques.

C'est en particulier ce qui est mis en avant dans [P. J. Liu *et al.*, 2018]. En effet, la tâche abordée ici consiste à générer l'introduction d'un article Wikipédia (appelé le *lead*) à partir des références de cet article, et d'informations du web. Pour ce faire, ils utilisent une architecture avec seulement un décodeur (pas d'encodeur donc), et présentent deux nouvelles couches : (i) *l'attention locale* (*local attention*) qui calcule l'attention sur des sous-séquences de l'entrée ; et (ii) *l'attention à mémoire compressée* (*memory compressed attention*) qui compresse l'information par convolution. L'objectif de ces deux mécanismes

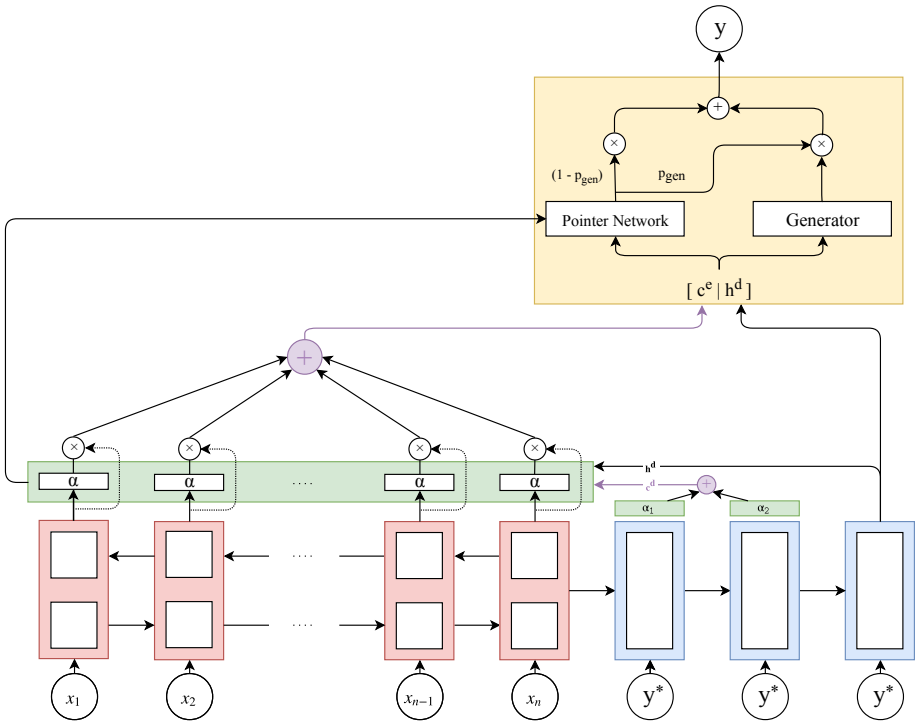


FIGURE 3.1 – Architecture du modèle de [SEE *et al.*, 2017] : un *encodeur-décodeur* (respectivement en rouge et en bleu) avec mécanisme d'attention (en vert). Le mécanisme d'attention côté encodeur prend en compte le vecteur couverture *cd* calculé à partir des attentions passées. Enfin, le mécanisme de *pointeur-générateur* (en jaune) combine les distributions obtenues par le *pointeur* (= mécanisme de copie) et du *générateur*

Encodeur-Décodeur		
Encodeur	Bi-LSTM, $\boldsymbol{h}_i^e =$	$\left[\overrightarrow{\boldsymbol{h}_i^e}^\top ; \overleftarrow{\boldsymbol{h}_i^e}^\top\right]^\top$, with $i \in [1; \lvert x \rvert]$
	$\overrightarrow{\boldsymbol{h}_i^e} =$	$lstm(\boldsymbol{h}_{i-1}^e; \boldsymbol{x}_i)$
	$\overleftarrow{\boldsymbol{h}_i^e} =$	$lstm(\boldsymbol{h}_{i+1}^e; \boldsymbol{x}_i)$
Décodeur	LSTM, $\boldsymbol{h}_t^d =$	$lstm(\boldsymbol{h}_{t-1}^d, y_t)$, with $\in [1; \lvert y \rvert]$
Attention		
Couverture	$\boldsymbol{c}_t^d =$	$\sum_{t'=0}^{t-1} \alpha_{t'}$
Énergie	$e_{ti} =$	$v^\top \tanh(\mathbf{W}_{attn}^e \boldsymbol{h}_t^d + \mathbf{V}_{attn}^e \boldsymbol{h}_i^e + \boldsymbol{c}_t^d + b_{attn})$
Distribution	$\alpha_t =$	$\text{softmax}(e_t)$
Contexte	$\boldsymbol{c}_t^e =$	$\sum_i \alpha_{ti} \boldsymbol{h}_i^e$
Pointeur-Générateur		
Générateur	$P_{generator} =$	$\text{softmax}(\mathbf{W}_{out}(\mathbf{V}_{out}\left[\boldsymbol{h}_t^d; \boldsymbol{c}_t^e\right] + b_{out'}) + b_{out})$
Taux Génération	$p_{gen} =$	$\sigma({w'_e}^\top \boldsymbol{c}_t^d + {w'_d}^\top \boldsymbol{h}_t^d + {w'_x}^\top \boldsymbol{x}_t)$
Distribution Finale	$P_{pointer\_generator} =$	$p_{gen} P_{generator}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_{ti}$

TABLE 3.1 – Description du modèle de [SEE *et al.*, 2017]

est de permettre au modèle de traiter efficacement un très grand nombre d'entrées : de l'ordre de 10^6 mots.

Ils proposent également d'effectuer d'abord une passe extractive en utilisant des méthodes statistiques telles que *TextRank* ou *TF-IDF* afin de sélectionner les portions de textes les plus pertinentes parmi la grande quantité de référence et de résultats de recherche web. Le résultat de l'extraction forme ensuite la source du modèle abstractif. Ils présentent des résultats largement supérieurs aux modèles *encodeur-décodeur* de références basés sur des RNN : 38.8 de ROUGE-L et 1.90 de perplexité contre 12.7 de ROUGE-L et 5.04 perplexité.

Leur jeu de données a été publié sont le nom `WikiSum`. Les entrées sont composées des références des articles Wikipédia et d'adresses de site web et dont les cibles sont les introductions des articles.

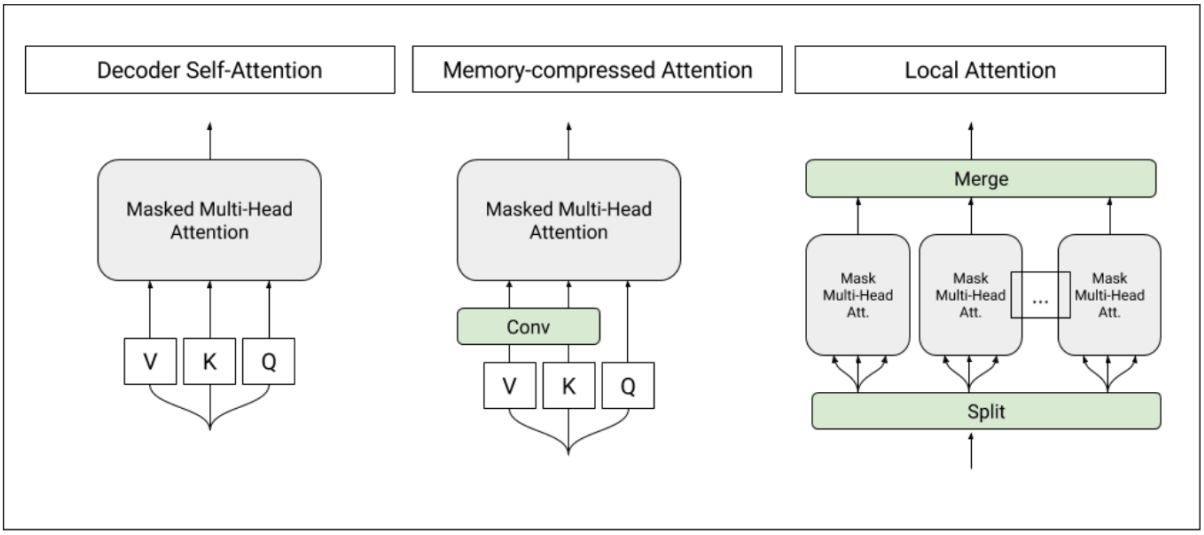


Figure 3.2 – Couches d'attention de [P. J. Liu *et al.*, 2018] (issu du papier original). L'architecture des couches d'auto-attention utilisées dans le modèle T-DMCA. Chaque couche d'attention prends une séquence de mots en entrée et produit une séquence de longueur similaire en sortie. **À gauche :** auto-attention originale du modèle Transformer.. **Au milieu :** attention à mémoire compressée permettant de réduire le nombre de clés/valeurs (resp. K, V). **À droite :** Attention locale, séparant les séquences en sous-séquences indépendantes. La séquence finale est ensuite obtenue en concaténant les sous-séquences.

3.4 Approche par Renforcement (RL)

3.4.1 Introduction à l'apprentissage par Renforcement pour la génération de texte

Les modèles *sequence-to-sequence* abordés jusqu'ici, qu'il s'agisse d'encodeur-décodeur à base de RNN (sections 2.3 et 3.1) ou TRANSFORMER (sections 2.4 et 3.3) sont entraînés à minimiser la *log-vraisemblance négative* des séquences cibles $\boldsymbol{y}$ de manière *autoregressive* :

$$\min_{\theta} -\frac{1}{|\boldsymbol{y}|} \sum_{t=1}^{|\boldsymbol{y}|} \log p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t}; \boldsymbol{x}) \tag{3.11}$$

Ce type d'apprentissage présente deux limites, présentées par [RANZATO *et al.*, 2016] :

1. **Biais d'exposition.** Les modèles de langage neuronaux et les décodeurs sont entraînés à prédire le mot $\boldsymbol{y}_t$ à partir des mots précédents $\boldsymbol{y}_{<t}$ et de l'entrée $\boldsymbol{x}$ (dans le cas des décodeurs) *i.e.* $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$.

   Par conséquent, le modèle n'accumule pas d'erreur dans ce cas, car même si sa prédiction est erronée à $t-1$ (*i.e.* $\hat{\boldsymbol{y}}_{t-1} \neq \boldsymbol{y}_{t-1}$ [2]) le modèle sera entraîné, au temps t à partir du résumé cible $\boldsymbol{y}_{<t}$ et non du résumé erroné $\hat{\boldsymbol{y}}_{<t}$. En d'autre termes, les erreurs à un instant n'impactent pas les instants suivants.

   Or, lors de la génération, c'est-à-dire en dehors du cadre de l'apprentissage, le modèle est livré à lui même : il ne dispose plus de la séquence cible $\boldsymbol{y}_t$. Le modèle effectue alors ses prédictions à partir des prédictions précédentes *i.e.* $p(\hat{\boldsymbol{y}}_t \mid \hat{\boldsymbol{y}_{t-1}}; \boldsymbol{x})$. Par conséquent, durant la génération, les erreurs s'accumulent : une erreur à l'instant $t-1$ viendra bruiter le contexte $\hat{\boldsymbol{y}}_{<t}$ et perturber la prédiction à l'instant t. Le modèle étant uniquement exposé aux séquences cibles $\boldsymbol{y}_{<t}$ (et non aux prédictions $\hat{\boldsymbol{y}}_{<t}$) durant l'apprentissage n'est pas robuste face à cette accumulation d'erreurs. Ce décalage entre le contexte d'apprentissage et de génération est appelée le *biais d'exposition*.

2. **Fonction de coût au niveau *mot*.** La fonction de coût des modèles de langage et des modèles encodeur-décodeur est exprimée de manière *autoregressive*, c'est-à-dire que la vraisemblance de la séquence est exprimée comme étant le produit des vraisemblances de chaque mot (= la somme des log-vraisemblances) *i.e.* $\mathcal{L}(\theta) =$

2. On note $\hat{\boldsymbol{y}}_{t-1}$ la prédiction du modèle au temps $t-1$, on par exemple $\hat{\boldsymbol{y}}_{t-1} = \arg\max p(\boldsymbol{y}_{t-1} \mid \hat{\boldsymbol{y}}_{<t-1}; \boldsymbol{x})$ dans le cas d'un décodeur glouton (*greedy decoding*).

$-\frac{1}{|\boldsymbol{y}|}\sum_{t=1}^{|\boldsymbol{y}|}\log p(\boldsymbol{y}_t \mid \boldsymbol{y}_{<t}; \boldsymbol{x})$. Or, la performance des modèles est mesurée au niveau des séquences entières, avec des métriques comme ROUGE ou BLEU par exemple, ce qui crée un décalage entre l'objectif de l'entraînement et celui de la génération.

Pour pallier ces deux problèmes, [Rennie *et al.*, 2017] proposent d'apprendre à minimiser directement la métrique de l'évaluation à partir de la séquence générée. En d'autres termes, ils se placent dès l'entraînement dans le même contexte que la génération. Toutefois, les métriques d'évaluation, comme ROUGE, ou CIDEr Vedantam *et al.*, 2015 dans le cas de Rennie *et al.*, 2017), ne sont pas dérivables. Leur approche s'appuie donc sur l'apprentissage par *Renforcement* (*Reinforcement Learning*) dont l'objectif est justement d'apprendre à maximiser une fonction non dérivable.

L'apprentissage par Renforcement consiste, pour un *agent* autonome dans un état $s_t \in \boldsymbol{\mathcal{S}}$, à déterminer l'*action* $a_t \in \boldsymbol{\mathcal{A}}$ qui maximise la *récompense* $r(s_t, a_t)$ au sein d'un *environnement*. On cherche alors à apprendre une *politique* permettant de déterminer l'action à prendre en fonction de l'état, c'est-à-dire la fonction $\pi : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \mapsto [0; 1]$, où $\pi(a_t \mid s_t)$ est la probabilité de prendre l'action a_t dans l'état s_t. Cette approche est notamment utilisée afin d'optimiser le comportement d'agents autonomes (*e.g.* robots) dans un environnement donné, ou à déterminer la meilleure stratégie de jeux (*e.g.* les échecs, le go, *etc.*). Dans le cadre du résumé automatique le modèle (= l'*agent*) interagit avec ses entrées x et $y_{<t}$ (= son *environnement*) et génère des mots (= *action*) grâce à la distribution de probabilité p_θ (= *politique*) calculée à partir des paramètres θ et de l'état du décodeur (= *état*).

En d'autres termes, on cherche les valeurs des paramètres θ tels qu'une séquence y^s échantillonnée à partir de la distribution p_θ obtienne la plus grande récompense $r(y^s)$ possible. Formellement, l'objectif de l'apprentissage est :

$$\min \boldsymbol{\mathcal{L}}^*(\theta) = -\mathbb{E}_{y^s \sim p_\theta}\left[r(y^s)\right] \tag{3.12}$$

La fonction de récompense r n'étant pas dérivable, l'algorithme Reinforce de [Williams, 1992] utilise une estimation des gradients $\nabla_\theta \boldsymbol{\mathcal{L}}(\theta)$ à partir de l'échantillon $y^s \sim p_\theta$:

$$\nabla_\theta \boldsymbol{\mathcal{L}}^*(\theta) = -\mathbb{E}_{y^s \sim p_\theta}\left[r(y^s)\nabla_\theta \log p_\theta(y^s)\right] \approx -r(y^s)\nabla_\theta \log p_\theta(y^s) \tag{3.13}$$

[Rennie *et al.*, 2017] utilisent l'algorithme Reinforce dans sa formulation *avec référence* où la récompense est évaluée par rapport à une valeur de référence (*baseline*) b c'est-à-dire que l'on remplace la récompense $r(y^s)$ dans l'équation 3.13 par la différence

avec la référence : $r(y^s) - b$. L'idée centrale de la contribution de [Rennie *et al.*, 2017] appelée *Self-Critical Sequence training* (*SCST*) est d'utiliser la récompense de la prédiction $\hat{y}$ (*i.e.* la sortie du modèle durant la génération) comme valeur de référence. On a alors :

$$\mathcal{L}^*(\theta) = -\mathbb{E}_{y^s \sim p_\theta}\left[r(y^s) - r(\hat{y})\right] \tag{3.14}$$

$$\nabla_\theta \mathcal{L}^*(\theta) \approx (r(\hat{y}) - r(y^s))\nabla_\theta \log p_\theta(y^s) \tag{3.15}$$

$$\hat{y}_t = \arg\max p_\theta(\hat{y}_t \mid \hat{y}_{<t})$$

On peut alors définir la fonction de coût $\mathcal{L}$ (approximation de $\mathcal{L}^*$) :

$$\begin{aligned}\mathcal{L}(\theta) &= (r(\hat{y}) - r(y^s)) \log p_\theta(y^s) \\ &= (r(\hat{y}) - r(y^s)) \sum_{t=1}^{N} \log p_\theta(y_t^s \mid y_{<t}; x)\end{aligned} \tag{3.16}$$

Minimiser cette fonction de coût $\mathcal{L}(\theta)$ (eq. 3.16) signifie maximiser les probabilités de y^s si sa récompense est supérieure à celle de $\hat{y}$, et les minimiser dans le cas contraire. Ceci revient donc à maximiser la récompense du modèle.

Dans la suite de cette section nous présentons d'abord le modèle de [Paulus *et al.*, 2017] (en section 3.4.2), un encodeur-décodeur basé sur des RNN dont les motivations sont similaires au PointerGen de (See *et al.*, 2017 *cf.* 3.2.4) et qui utilise l'apprentissage par Renforcement pour maximiser le score ROUGE du modèle. La section 3.4.3 présente d'autres fonctions de récompenses dans le cadre de l'entraînement par Renforcement de modèles Transformer pour le résumé automatique.

3.4.2 ROUGE comme fonction de récompense : le modèle de [Paulus *et al.*, 2017]

Les motivations de [Paulus *et al.*, 2017] sont similaires à celles de [See *et al.*, 2017] : l'exploration de techniques pour le résumé abstractif de longues séquences à partir d'un modèle encodeur-décodeur.

[Paulus *et al.*, 2017] utilisent deux mécanismes distincts pour traiter les problèmes de *couverture* (*i.e.* s'assurer que le résumé couvre bien tout le texte source, pas seulement une partie) et de *répétition*[3]. En premier lieu, l'attention temporelle (similaire à 3.2.1) est

3. En comparaison, le mécanisme de couverture de [See *et al.*, 2017] résout les deux problèmes directement.

utilisée pour limiter l'attention portée à chaque entrée, et ainsi accroître la couverture. Les auteurs observent cependant que ce mécanisme ne permet pas à lui seul de limiter la répétition. Ils introduisent donc une nouvelle couche d'attention *intra-décodeur* par opposition à l'attention classique, *intra-encodeur*. De même que l'attention *intra-encodeur* apprend à pondérer les entrées à chaque instant t, l'attention *intra-décodeur* prend en compte les précédents états du décodeur $\boldsymbol{h}^d_{<t}$. En particulier, ceci permet d'apprendre au décodeur à limiter les répétitions grâce à la comparaison de l'état courant avec les états précédents.

Il est également proposé de partager les poids du modèle entre les plongements de mots du décodeur et du générateur. L'idée est de mettre en commun l'apprentissage des deux couches dont le rôle est similaire : les plongements de mots associent chaque mot à un vecteur et le générateur associe le vecteur de sortie du réseau aux probabilités de chaque mot (en quelque sorte l'opération inverse).

L'architecture du modèle est représentée en figure 3.3 et les équations sont données dans le tableau 3.2.

La contribution centrale de [Paulus *et al.*, 2017] réside dans l'entraînement lui-même. En effet, ils suggèrent de combiner une fonction de coût $\boldsymbol{\mathcal{L}}_{ml}$ par maximum de vraisemblance (*maximum likelihood*, soit la fonction de coût communément utilisée pour cette tâche) avec une fonction $\boldsymbol{\mathcal{L}}_{rl}$ visant à maximiser le score ROUGE par Renforcement. Les deux fonctions de coût sont combinées au sein d'une fonction $\boldsymbol{\mathcal{L}}_{mixed} = \gamma\boldsymbol{\mathcal{L}}_{rl} + (1-\gamma)\boldsymbol{\mathcal{L}}_{ml}$ où γ est un hyper-paramètre du modèle. Ils utilisent *SCST* de [Rennie *et al.*, 2017] (*cf.* 3.4.1) avec pour récompense le score ROUGE_L par rapport à la référence $\boldsymbol{y}$ *i.e.* $r(\cdot) = \text{ROUGE}_L(\cdot, \boldsymbol{y})$. Pour chaque paire d'entraînement $(\boldsymbol{x}; \boldsymbol{y})$, deux passes de décodeurs sont exécutées afin d'obtenir les séquences y^s avec $y^s_t \sim p(y^s_t \mid y^s_{<t}; \boldsymbol{x})$ et $\hat{y}$ avec $\hat{y}_t = \arg\max p(y_t \mid y_{<t}, \boldsymbol{x})$ comme représenté fig. 3.4. Le coût par Renforcement ($\boldsymbol{\mathcal{L}}_{rl}$) est calculé à partir de la différence de récompense de y^s par rapport à $\hat{y}$ et des log-probabilités de y^s. Le coût $\boldsymbol{\mathcal{L}}_{ml}$ est calculé comme pour l'apprentissage classique d'un encodeur-décodeur : à partir des log-probabilités de y (voir détail du modèle tableau 3.2).

3.4.3 Alternatives à ROUGE comme fonction de récompense

Suite aux travaux de [Paulus *et al.*, 2017] d'autres fonctions de récompense ont été explorées pour l'apprentissage par renforcement du résumé automatique. En effet, plusieurs travaux de résumé abstractif [Paulus *et al.*, 2017 ; See *et al.*, 2017] observent les limites de la métrique ROUGE, qui favorise les résumés extractifs et n'est pas toujours

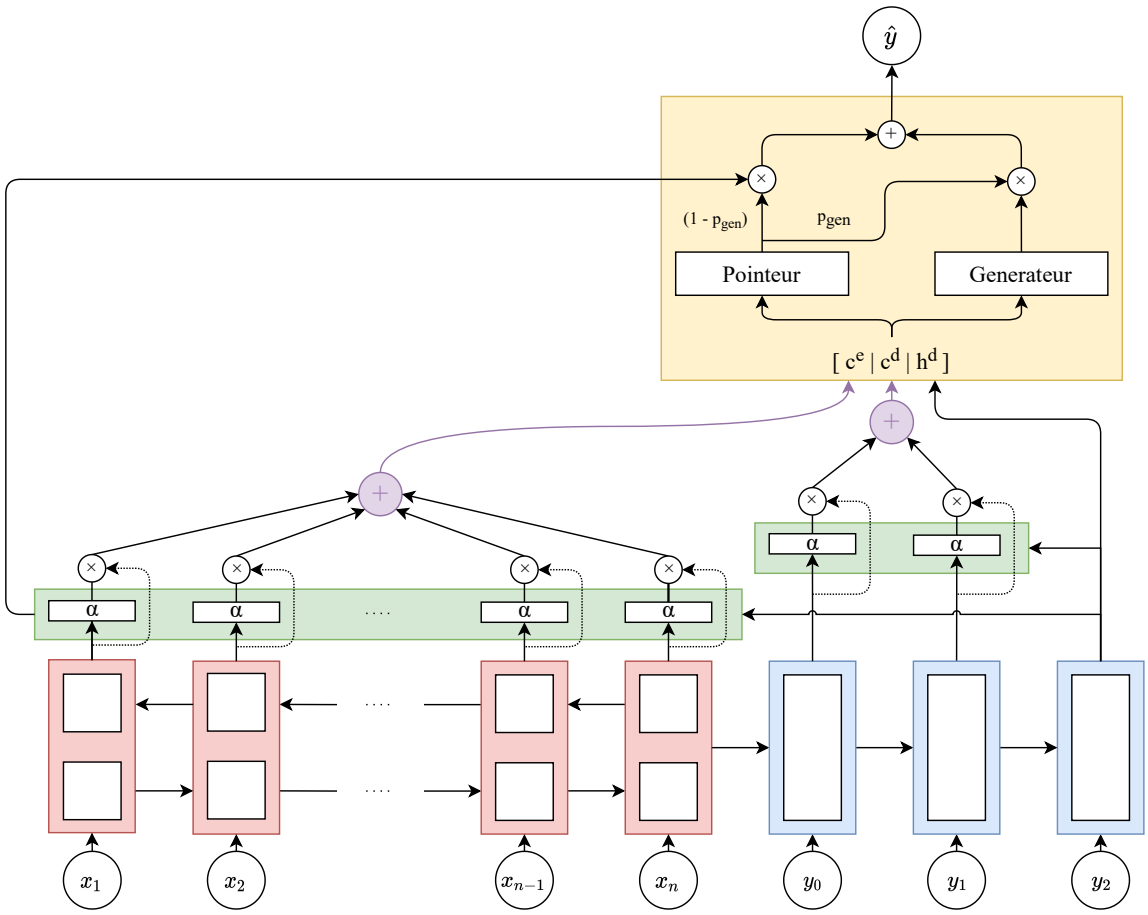


FIGURE 3.3 – Architecture du modèle de [PAULUS *et al.*, 2017] : un *encodeur-décodeur* (respectivement en rouge and en bleu) avec deux couches d'*attention* (vert), produisant chacune un *vecteur contexte* (violet) qui sont passés à la couche *pointeur-générateur* (jaune).

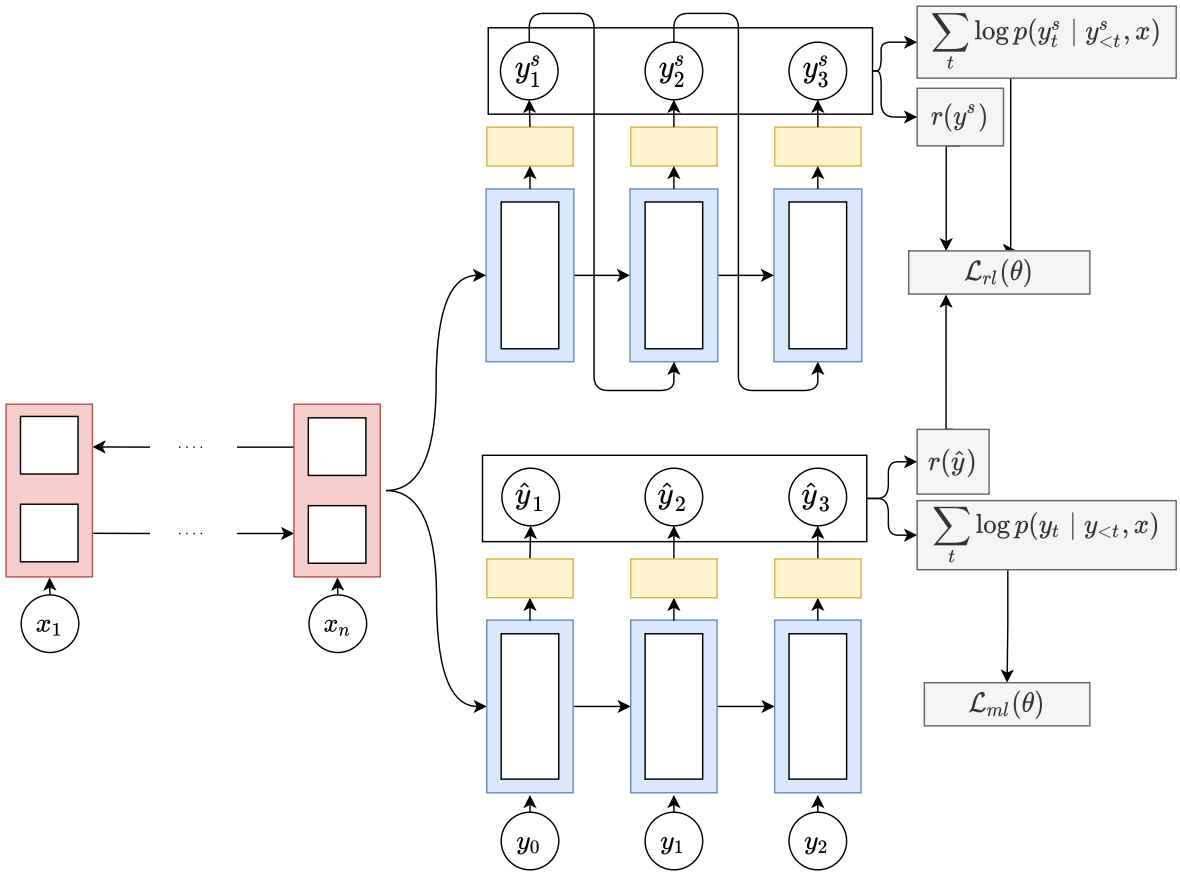


FIGURE 3.4 – Apprentissage mixte de [PAULUS *et al.*, 2017]. À partir de la sortie de l'encodeur (rouge), le décodeur (bleu) est exécuté en deux passes indépendantes : l'une (en bas) produit la *référence* $\hat{y}$ par *greedy-search* avec *teacher forcing* et permet de calculer les log-probabilités de la séquence cible y ; l'autre (en haut) pour obtenir y^s par échantillonage.

Encodeur-Décodeur		
Encodeur	$\boldsymbol{h}_i^e =$	$\left[\overrightarrow{\boldsymbol{h}_i^e}^\top ; \overleftarrow{\boldsymbol{h}_i^e}^\top\right]^\top, i \in [1; \lvert x\rvert]$
	$\overrightarrow{\boldsymbol{h}_i^e} =$	$lstm(\boldsymbol{h}_{i-1}^e; \boldsymbol{x}_i)$
	$\overleftarrow{\boldsymbol{h}_i^e} =$	$lstm(\boldsymbol{h}_{i+1}^e; \boldsymbol{x}_i)$
Décodeur	$\boldsymbol{h}_t^d =$	$lstm(\boldsymbol{h}_{t-1}^d, y_t), t \in [1; \lvert y\rvert]$
Attention Temporelle Intra-Encodeur		
Énergie :	$\boldsymbol{e}_{ti}^e =$	$\boldsymbol{h}_t^{d\top} \mathbf{W}_{attn}^e \boldsymbol{h}_i^e$
Énergie Temporelle :	$\boldsymbol{e}_{ti}^{tmp} =$	$\begin{cases} \exp(\boldsymbol{e}_{ti}^e) & \text{si } t = 1 \\ \frac{\exp(\boldsymbol{e}_{ti}^e)}{\sum_{j=1}^{t-1} \exp(\boldsymbol{e}_{ji}^e)} & \text{sinon} \end{cases}$
Attention Temporelle :	$\alpha_{ti}^e =$	$\frac{\boldsymbol{e}_{ti}^{tmp}}{\sum_{j=1}^{n} \boldsymbol{e}_{tj}^{tmp}}$
Contexte Encodeur :	$\boldsymbol{c}_t^e =$	$\sum_i \alpha_{ti}^e \boldsymbol{h}_i^e$
Attention Intra-Décodeur		
Énergie :	$\boldsymbol{e}_{tt'}^d =$	$\boldsymbol{h}_t^{d\top} \mathbf{W}_{attn}^d \boldsymbol{h}_{t'}^d$, with $t' \in [1; t-1]$
Attention :	$\alpha_{ti}^d =$	$\frac{\exp(\boldsymbol{e}_{ti}^e)}{\sum_{j=1}^{t-1} \exp(\boldsymbol{e}_{ji}^e)}$
Contexte Décodeur :	$\boldsymbol{c}_t^d =$	$\sum_i \alpha_{ti}^d \boldsymbol{h}_i^e$
Pointeur-Générateur		
Distribution Générateur :	$P_{gen} =$	$\text{softmax}(\mathbf{W}_{out} \left[\boldsymbol{h}_t^d \| \boldsymbol{c}_t^e \| \boldsymbol{c}_t^d\right] + b_{out})$
Taux génération :	$p_{gen} =$	$\sigma(W_u \left[\boldsymbol{h}_t^d \| \boldsymbol{c}_t^e \| \boldsymbol{c}_t^d\right] + b_u)$
Distribution copie :	$P_{copy}(w_t = x_i) =$	α_{ti}^e
Distribution Finale :	$P_{pointer\_gen}(w) =$	$p_{gen} P_{gen}(w) + (1 - p_{gen}) P_{copy}(w)$
Training Objectives		
Coût ML :	$\boldsymbol{\mathcal{L}}_{ml} =$	$-\sum_{t=1}^{\lvert y\rvert} \log p(y_t \mid y_1, \ldots, y_{t-1}; \boldsymbol{x})$
Coût RL :	$\boldsymbol{\mathcal{L}}_{rl} =$	$(r(\hat{y}) - r(y^s)) \sum_{t=1}^{\lvert y\rvert} \log p(y_t^s \mid y_1^s, \ldots, y_{t-1}^s; \boldsymbol{x})$
	où $r(\cdot) =$	$\text{ROUGE}_L(\cdot, y)$
Coût mixte :	$\boldsymbol{\mathcal{L}}_{mixed} =$	$\gamma \boldsymbol{\mathcal{L}}_{rl} + (1 - \gamma) \boldsymbol{\mathcal{L}}_{ml}$

TABLE 3.2 – Le modèle de [PAULUS *et al.*, 2017] en détails.

en adéquation avec le jugement humain.

[Scialom *et al.*, 2020b] utilisent des métriques issues de systèmes de *Questions-Réponses* (*Question Answering*, *QA*) : la qualité d'un résumé est mesurée comme sa capacité à apporter les réponses à des questions, par exemple l'identification d'entités nommées. Sont utilisées les métriques QA_F, à partir de la F-mesure du taux de réponses correctes, et QA_{conf} issue de la confiance du système de *Questions-Réponses*. Ces métriques sont combinées avec le score ROUGE : $r(\cdot) = \alpha\,\text{ROUGE}_L(\cdot, y) + \beta QA_{conf} + \delta QA_F$.

D'autres travaux proposent d'apprendre la fonction de coût directement à partir du jugement humain, [Böhm *et al.*, 2019 ; D. M. Ziegler *et al.*, 2019 ; Stiennon *et al.*, 2020]. Les contributions de ces deux travaux suivent trois axes : (i) l'évaluation humaine de résumés automatiques ; (ii) l'apprentissage de modèles de récompense à partir des évaluations humaines ; (iii) l'entraînement de modèles de résumé automatique par renforcement avec la fonction de récompense apprise. Leurs résultats montrent d'une part qu'il est possible d'apprendre des métriques sans références (le score est calculé uniquement à partir de l'entrée et du résumé automatique, pas de la cible) à partir de peu d'exemples (*e.g.* 2500 annotations) et dont la corrélation avec le jugement humain est meilleur que ROUGE ; et d'autre part que ces métriques permettent d'entraîner par renforcement des modèles de résumé abstractif.

Ils discutent également de deux problèmes quant aux évaluations de résumés automatiques :

— **Corrélation entre ROUGE et le jugement humain.** L'évaluation des métriques automatiques en termes de corrélation avec le jugement humain porte un regard critique sur ROUGE. En effet, ils observent que l'accord entre la métrique ROUGE et le jugement humain décroît à mesure que la qualité des résumés augmente. Par conséquent, évaluer la qualité des résumés uniquement d'après ROUGE peut mener à invalider des résumés (ou des modèles) pourtant meilleurs, ce qui est préoccupant.

— **Biais envers les résumés extractifs.** L'analyse des évaluations humaines révèle un biais des évaluateurs au profit des résumés extractifs. Ils constatent de la part des annotateurs une évaluation plus « conservatrice », c'est-à-dire bénéficiant aux résumés proches du document source, par rapport aux évaluations des chercheurs, qui récompensent davantage les reformulations. [Stiennon *et al.*, 2020] précisent avoir mis en place une formation et un suivi rigoureux de l'évaluation afin d'obtenir davantage de consensus entre annotateurs et chercheurs.

3.5 Spécialisation de modèles pour le résumé automatique : apprentissage par transfert et *fine-tuning*

Dans ce chapitre, nous avons présenté différentes méthodes neuronales pour l'apprentissage du résumé abstractif. Toutefois, toutes ces méthodes ont en commun l'approche choisie lors de l'entraînement : le réseau est entraîné à partir de zéro, de bout-en-bout et de manière supervisée sur la tâche du résumé abstractif. Or, on a vu lors du chapitre 2 que l'on peut entraîner des représentations génériques, c'est-à-dire indépendantes de la tâche finale afin de factoriser les temps de calculs, et maximiser la taille de jeux de données (qui ne sont plus alors contraints par la tâche). C'est en particulier le cas pour l'entraînement de représentations continues de mots, ou plongement de mots (*cf.* 2.2.1) ou même le pré-apprentissage d'encodeur ou de modèles de langages autorégressifs (*cf.* 2.5).

On appelle *apprentissage par transfert* le fait de transférer les connaissances acquises dans un contexte vers un autre contexte. Il peut s'agir d'une tâche différente, de l'application sur un jeu de données différent de celui de l'apprentissage ou bien de spécialisation d'un modèle sur une tâche plus spécifique. On distingue alors l'utilisation du modèle sans spécialisation (*zero shot*), c'est-à-dire sans aucun entraînement sur la tâche finale, du *fine-tuning* où le modèle est ré-entraîné sur la tâche finale. Certains travaux parlent également de *few shot learning* dans le cadre de *finetuning* sur un nombre très limité de données de spécialisation.

Certains modèles nécessitent d'apporter des modifications dans l'architecture entre le pré-entraînement et le *finetuning*. C'est notamment le cas de BertSumAbs[Y. Liu & Lapata, 2020] qui, basé sur Bert, ne dispose pas de décodeur lors de son pré-entraînement : celui-ci est alors ajouté entre le pré-entraînement et le *finetuning*. D'autres approches optent pour un modèle Transformer encodeur-décodeur dès le pré-entraînement [Lewis *et al.*, 2020 ; Raffel *et al.*, 2020 ; J. Zhang *et al.*, 2020] : le *finetuning* est alors effectué sur le modèle tel quel.

Cette approche en deux étapes – d'abord le pré-entraînement puis le *finetuning* – présente plusieurs avantages :

— **Mutualiser les données.** Le pré-apprentissage s'affranchit des contraintes spécifiques aux tâches, en particulier au besoin de disposer de paires d'entraînement alignées (par exemple de documents et de résumés). Ceci permet donc d'envisager le pré-entraînement sur un bien plus grand ensemble de données et de tirer profit de ces données sur plusieurs tâches.

— **Factoriser le temps de calcul.** L'expérimentation sur la tâche finale (*e.g.* le résumé automatique) est rendue moins coûteuse par le pré-entraînement : chaque expérience ne recommence pas l'apprentissage à partir de zéro.

— **Mieux généraliser.** Que ce soit par rapport à différentes tâches (*e.g.* traduction et résumé) ou bien différents jeux de données de la même tâche (*e.g.* données de CNN ou Ubiqus), les modèles issus de pré-entraînements de large échelle présentent de meilleures capacités de généralisation par rapport aux modèles supervisés classiques. Par exemple, les modèles de résumé automatique de *Bart* [Lewis *et al.*, 2020] et *Pegasus* [J. Zhang *et al.*, 2020] obtiennent de bonnes performances sur des données de journaux (*CNN/DailyMail* et *XSum*) ainsi que de Reddit (*ELI5* et *TIFU*).

3.6 Résumé *abstractif* ?

Au sein de ce chapitre, et dans cette thèse en général, nous nous intéressons au résumé *abstractif*. Nous avons distingué les approches *extractives* et *abstractives* en section 1.1 de la manière suivante :

> D'une part, l'approche *extractive*, consiste à résumer un document à partir de fragments de celui-ci. Cela revient à sélectionner (à extraire) les parties du document jugées les plus pertinentes.
> D'autre part, l'approche dite *abstractive*, consiste à rédiger le résumé de toutes pièces, à partir de l'analyse du document source. Cette approche n'est donc pas limitée au vocabulaire ni aux formulations du document source et permet donc un meilleur niveau de reformulation et de synthèse.

À partir de cette définition, on peut qualifier d'*abstractif* un résumé (ou le modèle qui le génère) dès lors qu'il contient au moins un mot n'étant pas présent dans le document source. Les modèles abstractifs sont entraînés dans ce but, soit par maximisation de la vraisemblance des mots y_t de la cible soit par renforcement à partir d'une séquence échantillonnée y^s. Les deux objectifs sont abstractifs dans la mesure où y_t n'est pas nécessairement présent dans x, donc l'apprentissage comprend la prédiction de mots absents de la source. De même, l'apprentissage par renforcement échantillonne la séquence y^s sur l'intégralité du vocabulaire, pas seulement sur les mots présents dans la source.

Toutefois, dès les premiers modèles neuronaux de résumé abstractif (*cf.* Rush *et al.*, 2015 sec. 3.1 ; See *et al.*, 2017 sec. 3.2.4) les auteurs constatent un *biais d'extractivité*, c'est-à-dire la tendance des résumés générés à être bien plus *extractifs* que les résumés de référence. En particulier, le caractère *extractif* d'un résumé peut être mesuré à partir de

la proportion de n-grammes du résumé étant également présents dans le document source (donc en quelque sorte *copiés* à partir de la source). Ils constatent alors des taux de copies deux fois supérieurs au sein des résumés automatiques par rapport aux références. Ce phénomène s'accroît davantage lorsque l'on compare la copie de bigrammes, trigrammes, etc. En cela, la notion d'*abstractivité* n'a pas vraiment de sens dans l'absolu, mais s'évalue par rapport à une référence.

De plus, de nombreux travaux sur le résumé *abstractif* observent des gains de performances lors de l'ajout de mécanismes *extractifs*. Ceci inclut :

— **Caractéristiques Extractives (*extractive features*).** Le modèle ABS+ de [RUSH *et al.*, 2015] accroît le comportement extractif du modèle de base ABS. Ceci ce traduit par un gain marginal de score ROUGE (+1 R1) et d'une hausse du taux de copie de 85.4% à 91.5% alors que le taux de copie de référence est de 45.6%.

— **Le mécanisme de copie.** Ce mécanisme permet de copier des mots de la source dans le résumé, ce qui permet en particulier de gagner en performance sur les mots rares, qu'il est plus aisé de copier que de générer. Toutefois, ceci peut amener les modèles à converger vers un comportement très extractif. Dans leurs questionnement sur l'abstractivité de leur modèle, [SEE *et al.*, 2017] rapportent que le taux de génération [4] se situe autour 50% lors de l'entraînement et baisse à moins de 17% lors de la génération ce qui traduit un fort biais d'extractivité.

— **Passe extractive, puis passe abstractive.** Avec BOTTOM-UP, [GEHRMANN *et al.*, 2018] proposent un fonctionnement en deux passes : un modèle prédit d'abord pour chaque mot de la source si celui-ci est pertinent ou non pour la copie. La deuxième passe introduit un modèle neuronal abstractif dont le mécanisme de copie est conditionné sur le résultat de la première passe. Le modèle neuronal ne peut alors copier que les mots préalablement autorisés, ce qui limite notamment la copie de longue portion de texte. Dans une idée similaire, [P. J. LIU *et al.*, 2018] procède d'abord à l'extraction de texte à partir d'une grande quantité de texte avant d'utiliser ce résultat comme source du modèle abstractif.

On peut alors se questionner sur l'objectif de la tâche lorsque l'on constate que l'augmentation des performances (en termes de score ROUGE) s'accompagne par une baisse

4. Dans un modèle avec mécanisme de copie, le taux de génération p_{gen} correspond à la pondération entre la distribution du générateur (*i.e.* le comportement abstractif) et la distribution de copie (*i.e.* comportement extractif). Une valeur faible correspond donc à un comportement essentiellement extractif. Ce mécanisme est présentée en détail section 3.2.3.

de l'abstractivité. Il semble que nous ayons deux objectifs qui s'opposent : l'abstractivité d'une part, et la similarité avec la référence (exprimée par le score ROUGE) d'autre part. Ce constat est d'autant plus préoccupant que la métrique ROUGE elle même tend à pénaliser les approches abstractives. En effet, ROUGE mesure les co-occurrences de n-grammes entre un résumé et sa référence. Ceci a pour conséquence de pénaliser les paraphrases, les synonymes ou les changements de formulation : autant d'éléments pourtant intéressants dans un résumé abstractif.

Le score ROUGE ayant montré ses limites, il est maintenant courant de procéder à une évaluation humaine des comptes rendus automatiques, en plus de l'évaluation automatique. Toutefois, l'évaluation humaine n'est pas parfaite non plus, et peut également pénaliser l'abstractivité. C'est en tout cas les résultats des travaux de [D. M. Ziegler *et al.*, 2019 ; Stiennon *et al.*, 2020] qui observent la présence d'un biais d'extractivité jusque dans le jugement humain : les annotateurs ont tendance à favoriser les résumés extractifs. Ils formulent l'hypothèse d'une évaluation plutôt *conservatrice* de la part des annotateurs. Les résumés plus proches de leur source pouvant sembler à première vue plus pertinents et sont moins prompts aux tournures hasardeuses ou imprécisions.

Pour conclure, la notion de résumé *abstractif* est assez difficile à définir, à apprendre et à mesurer, que ce soit grâce à des métriques ou en recourant à l'évaluation humaine.

3.7 Conclusion

Nous avons présenté dans ce chapitre l'état de l'art en matière d'approches neuronales pour le résumé abstractif de texte. Si les architectures de modèles neuronaux de résumé, comme l'*encodeur-décodeur*, sont communes à d'autres domaines de génération du langage – comme la traduction automatique – nous avons présenté des extensions à ces modèles particulièrement intéressantes dans le cadre du résumé automatique. En particulier, nous avons présenté les concepts d'*attention temporelle* et de *couverture* (section 3.2.1) permettant de traiter les problèmes de répétitions ainsi que le mécanisme de copie (aussi appelé *pointeur-générateur*) permettant de générer des mots rares (section 3.2.3). De même, nous avons présenté les travaux appliquant le modèle Transformer au résumé (section 3.3). Nous avons également abordé les motivations et les techniques d'apprentissage par renforcement (section 3.4.3) ainsi que le *fine-tuning* de modèles pré-entraînés (section 3.5), faisant notamment écho à la section 2.5 du chapitre précédent sur l'apprentissage auto-supervisé. Nous concluons ce chapitre par une discussion quant à l'abstractivité au sein des résumés,

en mettant en avant la difficulté à définir et évaluer cette notion.

Chapitre 4

Alignement des comptes rendus avec les transcriptions

L'apprentissage de réseaux de neurones de manière supervisée nécessite de disposer d'un ensemble de données constituée d'une part de *documents* et d'autre part, des *résumés* correspondants, ou respectivement des *transcriptions* et des *comptes rendus* dans notre contexte. On dit que les données sont *alignées* lorsque l'on dispose de paires d'entraînement, c'est-à-dire de paires *(entrée du réseau ; sortie souhaitée)*.

L'alignement de données est une étape capitale dans la constitution d'un ensemble de données pour l'apprentissage supervisé. Dans le cadre des comptes rendus de réunion, on traite chaque prise de parole et on cherche à aligner la transcription de cette prise de parole avec la section du compte rendu correspondant.

Dans ce chapitre, nous présentons notre approche concernant l'*alignement* des données Ubiqus grâce à des annotateurs humains dans un premier temps, puis proposons une méthodologie pour l'apprentissage de l'alignement automatique.

Table des matières

4.1 Problématique

Les données dont dispose UBIQUS, présentées en section 1.2 ont été segmentées (voir section 1.2.3 de manière à obtenir des textes de longueur satisfaisante en vue de l'entraînement de réseaux de neurones. Toutefois, il est nécessaire d'*aligner* les segments entre eux, c'est-à-dire d'associer à chaque segment du compte rendu les segments de la transcription correspondants.

En d'autres termes, nous pouvons définir une transcription $\boldsymbol{\mathcal{T}} = \left\{t_1, \ldots, t_{|\mathcal{T}|}\right\}$ et le compte rendu correspondant $\boldsymbol{\mathcal{R}} = \left\{r_1, \ldots, r_{|\mathcal{R}|}\right\}$ en tant qu'ensemble de phrases. D'autre part, nous disposons des ensembles de segments pour la transcription $\hat{\boldsymbol{\mathcal{T}}} = \left\{T_1, \ldots, T_{|\hat{\mathcal{T}}|}\right\}$, et le compte rendu $\hat{\boldsymbol{\mathcal{R}}} = \left\{R_1, \ldots, R_{|\hat{\mathcal{R}}|}\right\}$. Les segments sont des ensembles mutuellement exclusifs de phrases.

Dans notre cas, nous considérons que l'alignement consiste à trouver une application $\textit{alignement} : \hat{\boldsymbol{\mathcal{R}}} \mapsto \boldsymbol{\mathcal{P}}\left(\hat{\boldsymbol{\mathcal{T}}}\right)$ qui associe chaque segment compte rendu à un ensemble de segments transcription.

Comme détaillé en section 1.2.1, nous travaillons avec des comptes rendus dits « exhaustifs » ce qui implique :

— **Fidélité :** le compte rendu est fidèle à la réunion : le compte rendu ne traite que de propos prononcés au cours de la réunion (eq. 4.1).

$$\forall R_m \in \hat{\boldsymbol{\mathcal{R}}}, \textit{alignement}(R_m) \neq \varnothing \tag{4.1}$$

— **Exhaustivité :** tous les propos sont retranscrits dans le compte rendu (eq. 4.2).

$$\forall T_i \in \hat{\boldsymbol{\mathcal{T}}}, \exists R_m \in \hat{\boldsymbol{\mathcal{R}}}, T_i \in \textit{alignement}(R_m) \tag{4.2}$$

— **Chronologie :** le compte rendu conserve la chronologie de la réunion (eq. 4.3).

$$\begin{aligned} &\forall T_i \in \hat{\boldsymbol{\mathcal{T}}}, T_i \in \textit{alignement}(R_m), \\ &\forall T_j \in \hat{\boldsymbol{\mathcal{T}}}, T_j \in \textit{alignement}(R_n), \\ &i < j \Rightarrow m <= n \end{aligned} \tag{4.3}$$

— **Exclusivité :** chaque prise de parole est traitée indépendamment, et n'est associée

qu'à un unique segment de compte rendus (eq. 4.4).

$$\forall T_i \in \hat{\boldsymbol{\mathcal{T}}}, \forall (R_n, R_m) \in \hat{\boldsymbol{\mathcal{R}}}^2$$
$$\begin{cases} T_i \in \textit{alignement}(R_n) \\ T_i \in \textit{alignement}(R_m) \end{cases} \Rightarrow n = m \tag{4.4}$$

Ces propriétés du compte rendu exhaustif nous informent sur les caractéristiques souhaitées de l'alignement, ainsi que sur la manière de procéder. Par exemple, on suppose que les segments *transcription* alignés avec un segment *compte rendu* R_m sont contigus, c'est-à-dire que l'on peut écrire $\textit{alignement}(R_m) = A = \{T_i, \ldots, T_{i+k}\}$ (avec $k = |A|$). En d'autre termes, aligner le segment R_m revient à trouver le premier segment *transcription* T_i auquel il correspond, et le dernier T_{i+k} et à les sélectionner, ainsi que tous les segments intermédiaires.

Une fois le segment *compte rendu* R_m aligné, ainsi que les segments *transcription* $T_i, \ldots, T_{i+k}$, on sait, par *fidélité* (eq. 4.1) que le segment *compte rendu* suivant, R_{m+1}, doit être aligné à au moins un segment *transcription*. Par *exhaustivité* (eq. 4.2), on sait que le segment *transcription* suivant, T_{i+k+1} doit être aligné à exactement un segment *compte rendu*. Cet alignement est forcé, par *chronologie*, on a : $T_{i+k+1} \in \textit{alignement}(R_{m+1})$

On sait aussi, par *chronologie* (eq. 4.3), que R_{m+1} est nécessairement aligné avec le segment *transcription* suivant, c'est-à-dire $T_{i+|A|}$. On cherche alors le dernier segment *transcription* qui correspond, et ainsi de suite.

Par *fidélité* et *exhaustivité*, on sait aussi que $T_1 \in \textit{alignement}(R_1)$ et que $T_{\left|\hat{\boldsymbol{\mathcal{R}}}\right|} \in \textit{alignement}(T_{\left|\hat{\boldsymbol{\mathcal{T}}}\right|})$. L'alignement revient donc trouver les derniers segments *transcription* de $\left|\hat{\boldsymbol{\mathcal{R}}}\right| - 2$ segments *compte rendu* (le premier et le dernier sont exclus).

4.2 Alignement Humain

Afin d'obtenir un jeu de données pour l'apprentissage du résumé abstractif de réunion à partir des données segmentées dont nous disposons (voir 1.2.3) nous procédons à l'alignement des segments comptes rendus et des transcriptions grâce à des annotateurs humains.

Comme discuté en section précédente (4.1), l'alignement consiste à déterminer, pour chaque segment compte rendu, dans l'ordre chronologique, le dernier segment transcription qui correspond (le premier étant déterminé implicitement à la suite du segment précédent). Cette méthode est particulièrement pratique pour les annotateurs humains

car elle permet de ne pas lire l'intégralité des textes, mais de se contenter de trouver le segment transcription correspondant à la fin du segment compte rendu.

Pour cela, nous avons développé une plateforme *web* permettant aux annotateurs de charger les segments d'une réunion, disposés en deux colonnes, l'une pour les segments du compte rendu, l'autre pour les segments de la transcription (voir figure 4.1). Après avoir sélectionné un segment compte rendu, l'annotateur peut sélectionner autant de segments transcriptions que nécessaire, en cliquant dessus, ce qui va les ajouter à l'alignement. Un code couleur est là pour aider l'annotateur à se repérer. Les figures 4.1a et 4.1b présentent un exemple de cette procédure. Une fois tous les segments alignés, l'annotateur peut envoyer son travail, qui sera sauvegardé côté serveur.

Le serveur dispose d'une liste de réunions segmentées qui sont envoyées aux annotateurs à la demande – lorsque celui-ci charge une réunion depuis la plateforme. Les réunions sont annotées par un unique annotateur. Afin de simplifier la tâche d'annotation manuelle, la réunion est pré-alignée grâce à l'alignement automatique, qui est développé en section 4.3.

4.3 Alignement Automatique

L'alignement, tel que décrit en section 4.1 peut être envisagé automatiquement. En particulier, nous proposons de définir l'alignement automatique comme la recherche de l'application *alignement* qui associe à chaque segment du compte rendu un ou plusieurs segments de la transcription de manière à maximiser la similarité entre le segment du compte rendu et les segments de la transcription alignées.

Afin d'affiner la granularité des similarités – les segments pouvant être longs – nous calculons les similarités au niveau des phrases, donc, pour chaque phrase r_i du compte rendu $\boldsymbol{\mathcal{R}}$ et chaque phrase t_j de la transcription $\boldsymbol{\mathcal{T}}$ nous avons la similarité $S_{i,j} = score(r_i, t_j)$. L'objectif est ensuite, de calculer le meilleur alignement au niveau des segments, à partir des similarités aux niveaux des phrases.

Les représentations du texte et les fonctions de similarité des phrases (*i.e.* la fonction *score*) sont introduites en section 4.3.1. La section 4.3.2 présente le calcul de l'alignement par maximisation de la similarité entre segments alignés. Nous proposons en section 4.3.3 une approche itérative permettant d'une part de rendre l'annotation humaine plus simple au fil des itérations grâce au perfectionnement des modèles d'alignement automatique ; et d'autre part d'évaluer plus finement les modèles grâce à l'annotation de données de

FIGURE 4.1a – Plateforme d'alignement (1/2). L'annotateur cherche à aligner le segment courant (en gris). En particulier, on voit qu'un segment transcription est injustement aligné. En effet, le phrase : « par rapport à la cartographie... » doit être alignée avec le segment compte rendu suivant (en bleu). L'annotateur décochera alors ce segment transcription, et passera au segment compte rendu suivant (voir 4.1b).

FIGURE 4.1b – Plateforme d'alignement (2/2). Les segments précédents sont désormais validés (en vert) et on aligne le segmente document courant (en gris, à gauche) avec le segment transcription (en gris, à droite) et on désaligne le segment transcription suivant (en rouge, à droite) afin de pouvoir ensuite l'aligner avec le segment compte rendu suivant (en bleu, à gauche).

référence.

Enfin, en section 4.3.4, nous explorons trois approches pour l'évaluation de nos méthodes d'alignement automatique : (i) en proposant un parallèle entre alignement automatique et segmentation linéaire ; (ii) en utilisant les corrections des annotateurs (*i.e.* on considère que la performance du modèle est inversement proportionnelle au nombre de corrections qui ont été effectuées) ; (iii) en étudiant l'impact des données supplémentaire issues de l'alignement automatique sur les performances des modèles de résumé automatique.

4.3.1 Représentation du texte et calcul de similarité

La première étape de l'alignement automatique est d'obtenir une représentation du texte qui permette le calcul de similarité entre les phrases du compte rendu $\boldsymbol{\mathcal{R}}$ et les phrases de la transcription $\boldsymbol{\mathcal{T}}$.

On cherche alors à construire une matrice de similarité S entre $\boldsymbol{\mathcal{T}}$ et $\boldsymbol{\mathcal{R}}$ telle que $S_{i,j} = score(t_i, r_j)$ avec $(t_i, r_j) \in \boldsymbol{\mathcal{T}} \times \boldsymbol{\mathcal{R}}$.

Les choix de la fonction *score* et de la représentation du texte sont liés. En effet, on peut considérer la similarité entre deux séquences de manière textuelle, c'est-à-dire en étudiant directement les mots ; mais aussi de manière discrète, c'est-à-dire en dénombrant les occurrences des mots, alors représentés par des entiers ; et enfin de manière vectorielle, à partir de vecteurs de mots, par exemple des plongements de mots (voir section 2.2.1).

Largement utilisé dans le domaine du résumé automatique de texte, le score ROUGE (Lin, 2004 *cf.* section 1.3.1) compare deux textes sur la base de leur n-grammes communs. Les variantes les plus utilisées sont ROUGE-1, ROUGE-2 et ROUGE-L qui mesurent respectivement les proportions d'unigrammes, bigrammes et la longueur de la plus longue sous séquence commune aux deux textes.

Nous explorons aussi une représentation des mots sous forme de vecteurs TF-IDF. La mesure TF-IDF (*Term Frequency - Inverse Document Frequency*) a pour objectif de mesurer l'*importance* d'un mot, par sa fréquence au sein du document (c'est le TF) par rapport à sa fréquence d'apparition au sein des autres documents (c'est le IDF). L'intuition est que plus un mot est utilisé au sein d'un document (TF), plus il doit être *important*. Toutefois, en ne se basant que sur la fréquence des mots, on pourrait conclure que des mots courants comme « je », « le » ou « de » sont *importants* ce qui ne semble pas très pertinent. Or on observe que ces mots ne sont pas spécifiquement présents dans un document mais qu'au contraire ils sont très courants dans l'ensemble des documents

(IDF) : on les considère alors moins importants. En d'autres termes, un mot est considéré *important* s'il est fréquent au sein d'un document mais relativement rare dans les autres. On représente alors un texte par des vecteurs dont les dimensions représentent les mots qui le constitue et dont les valeurs sont les scores TF-IDF.

Nous utilisons également les plongements de mots (voir 2.2.1) afin d'obtenir une représentation vectorielle des mots. Les vecteurs ont été entraînés par [Fauconnier, 2015] avec word2vec [Mikolov *et al.*, 2013a] d'après les modèles CBOW et Skip-Gram (2.2.1).

La similarité cosinus est utilisée comme fonction *score* à partir des représentations vectorielles (par TF-IDF ou plongement de mots).

La représentation vectorielle d'une phrase est obtenue à partir des représentations de chacun de ses mots grâce à une fonction de *pooling*, qui peut être aussi simple qu'une somme. L'objectif du calcul des similarités entre les phrases des deux textes *(compte rendu, et transcription)* est d'identifier les passages où les deux textes traitent du même sujet. Dans ce contexte, il n'est pas trivial d'identifier le niveau de granularité pertinent pour mesurer la similarité. Si l'on a supposé que les longueurs des segments étaient trop grandes et trop peu uniformes, on peut se demander si se placer au niveau des phrases est plus pertinent. Afin d'élargir nos expériences, nous introduisons de nouveaux niveaux de granularité intermédiaires sous forme de fenêtres glissantes. Ainsi, pour tout document $D \in \{\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}\}$, la k-ième fenêtre glissante $W^D_{o,s}(k)$ est l'ensemble des s phrases dont les o dernières phrases (respectivement, premières) sont partagées avec la fenêtre suivante, la $(k+1)$-ième (resp. fenêtre précédente, la $(k-1)$-ième).

$$W^D_{o,s}(k) = \left\{ s_{ks-ko}, ..., s_{(k+1)s-ko} \right\} \in D^s \tag{4.5}$$

Une fonction *d'agrégation* – notée *agg* – calcule les représentations des fenêtres glissantes à partir des représentations des phrases qui la composent (voir figure 4.2). Dans nos expériences nous utilisons comme fonction d'agrégation la somme (*sum*), la moyenne (*mean*), la valeur maximum (*max*) ou la concaténation (*cat*, pour les séquences de textes). On calcule ensuite la matrice de similarité au niveau des fenêtres glissantes :

$$S^{sliding}_{k,l} = score\left(agg(W^T_{o,s}(k)), agg(W^R_{o,s}(l))\right) \tag{4.6}$$

Les similarités entre les fenêtres sont ensuite assignées aux phrases les composant. Les phrases pouvant faire partie de plusieurs fenêtres, et donc avoir un ensemble de scores (au

niveau des fenêtres), on réduit cet ensemble – avec la fonction red – de sorte à obtenir un unique score pour toute paire de phrases (t_i, r_j) dans $\boldsymbol{\mathcal{T}} \times \boldsymbol{\mathcal{R}}$ (voir eq. 4.5).

$$S_{i,j} = red\left(\left\{S_{k,l}^{sliding} \mid (t_i, r_j) \in \left(W_{o,s}^{T}(k) \times W_{o,s}^{R}(l)\right)\right\}\right) \tag{4.7}$$

La fonction de réduction red est typiquement une somme ou un produit.

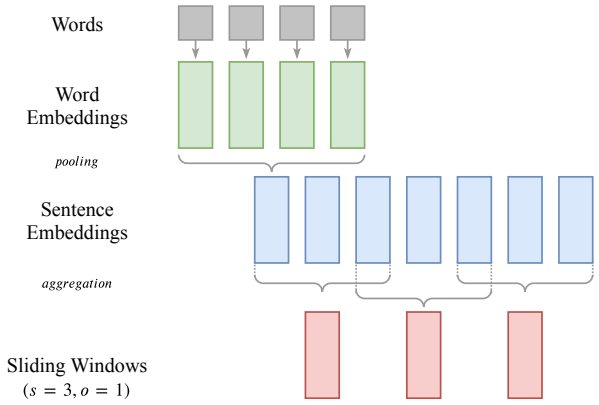


FIGURE 4.2 – Représentation du texte dans le cadre des approches vectorielles de l'alignement automatique.

4.3.2 Calcul de l'alignement

Ayant calculé les similarités entre toutes les paires de phrases de la transcription et du compte rendu, la tâche de l'alignement consiste maintenant à maximiser la similarité au niveau du document. Nous utilisons la programmation dynamique afin de trouver un chemin optimal dans la matrice de similarité S.

Soit une matrice d'alignement A dont la valeur correspond, pour chaque coordonnée (i, j), à la somme de la similarité $S_{i,j}$ (éventuellement élevée à la puissance p) et de la valeur maximale entre les valeurs aux coordonnées $(i, j-1)$ et $(i-1, j)$ (eq. 4.8).

$$\begin{aligned} A_{i,j} &= {S_{i,j}}^{p} + \max(A_{i-1,j}, A_{i,j-1}) \\ A_{i,0} &= 0 \\ A_{0,j} &= 0 \end{aligned} \tag{4.8}$$

On peut voir la construction d'une telle matrice comme un signal qui se propage, à

partir de l'angle en haut à gauche $(1, 1)$, jusqu'à l'angle en bas à droite $(|\boldsymbol{\mathcal{T}}|, |\boldsymbol{\mathcal{R}}|)$ de proche en proche, de sorte à maximiser la similarité le long de son trajet.

À chaque position (i, j) on garde la trace de la position précédente, c'est-à-dire de la position $(i-1, j)$ ou $(i, j-1)$) dont la valeur d'alignement (dans A) est maximale.

$$H_{i,j} = \underset{c \in \{(i-1,j);(i,j-1)\}}{\arg\max} (A_c) \tag{4.9}$$

On peut alors construire le chemin optimal $P = \{c_1, \ldots, c_N\}$ de $N = |\boldsymbol{\mathcal{T}}| + |R|$ coordonnées, à partir des dernières coordonnées $c_N = (|\boldsymbol{\mathcal{T}}|; |R|)$ en remontant l'historique H itérativement :

$$\begin{aligned} &\forall n \in \{1, N\} \\ &c_n = H_{i,j} \text{ avec } (i, j) = c_{n+1} \\ &c_N = (|\boldsymbol{\mathcal{T}}|, |\boldsymbol{\mathcal{R}}|) \end{aligned}$$

La figure 4.3 montre un exemple d'alignement à partir d'une matrice de similarité S.

L'alignement au niveau des segments découle de l'alignement au niveau des phrases. Un segment transcription T_n est aligné avec le segment compte rendu R_m qui maximise le score d'alignement similarité le long du chemin optimal P (eq. 4.10).

$$\textit{alignement}(R_m) = \left\{ T_i \mid R_m = \underset{R_j \in \hat{\mathcal{R}}}{\arg\max} \left(\sum_{s_k \in T_i} \sum_{s_l \in R_j} A_{k,l} \mathbb{1}_{(k,l) \in P} \right) \right\} \tag{4.10}$$

Dans l'équation 4.10 on définit *alignement* de R_m comme l'ensemble des segments transcriptions T_i pour lesquels R_m est le segment compte rendu qui maximise le score d'alignement le long du chemin optimal. Ce score est la somme des valeurs des scores $A_{k,l}$, pour chaque paire de phrases k, l issues respectivement du segment transcription et du segment compte rendu. Seules les paires de phrases faisant partie du chemin optimal sont considérées *i.e.* si $(k, l) \in P$.

L'alignement automatique par programmation dynamique garantit, par son fonctionnement, le respect des propriétés d'exhaustivité, exclusivité, chronologie énoncées en début de chapitre (4.2 à 4.3).

En effet, le parcours par programmation dynamique de la matrice de similarité débute aux coordonnées $(0, 0)$ (c'est-à-dire le point en haut à gauche de la matrice) et se termine au point $(|\boldsymbol{\mathcal{T}}|, |\boldsymbol{\mathcal{R}}|)$ (en bas à droite). Ainsi, le chemin optimal P passe par chaque phrase

de chaque segments transcription et compte rendu. Par conséquent, tout segment transcription T_i est aligné à un unique segment R_m (*exhaustivité* et *exclusivité*) d'après 4.10 [1]. De même, il n'existe pas de « saut » au sein du chemin optimal P : celui-ci progresse de 1 coordonnée, soit vers la droite, soit vers le bas, sans revenir en arrière ce qui garantie la *chronologie* de l'alignement. Seule la fidélité, c'est-à-dire le fait que chaque segment compte rendu soit aligné avec au moins un segment transcription n'est pas respecté. En effet, pour un segment R_m, il peut n'exister aucun segment transcription T_i tel que R_m soit solution du arg max de l'équation 4.10. En pratique, lever cette contrainte permet d'ignorer certains segments du compte rendu qui portent davantage sur la structure du document que sur la discussion de la réunion. C'est notamment le cas de la liste des personnes présentes ou non (qui est rarement présente dans la transcription), de titres du document ou d'informations supplémentaires.

	t_1	t_2	t_3	t_4
r_1	5	3	8	9
r_2	5	7	6	2
r_3	3	4	7	5

	t_1		t_2		t_3		t_4
r_1	5	→	8	→	16	→	25
	↓						↓
r_2	10	→	17	→	23		27
	↓		↓		↓		
r_3	13		21		30	→	35

FIGURE 4.3 – Exemple d'algorithme de programmation dynamique pour trouver l'alignement qui maximise la similarité. Pour toute coordonnée (i, j), l'alignement (A, à droite) ajoute à la similarité $S_{i,j}$ (à gauche, les valeurs sont arbitraires) la valeur d'alignement maximale parmis les coordonnées voisines, du haut $(i, j-1)$ ou de gauche $(i-1, j)$, comme représenté par les flèches (qui représentent l'historique H) ; Les flèches rouges représentent le chemin optimal P.

4.3.3 Approche itérative

Pour aligner le corpus de manière efficace, nous avons procédé itérativement autour de trois phases : (i) le pré-alignement des comptes rendus et transcriptions grâce aux modèles d'alignement automatique ; (ii) la correction des pré-alignements via l'annotation humaine (4.2) afin d'obtenir un nouveau jeu de données de référence ; (iii) l'évaluation des modèles d'alignement automatique par rapport aux données alignées de référence, permettant de

1. L'inverse supposerait que la fonction arg max de 4.10 ne retourne rien, ce qui est absurde.

mieux comparer les modèles et donc de mieux choisir le modèle pour les pré-alignements suivants.

Nous augmentons ainsi, à chaque itération, la quantité de données de référence, alignées par des annotateurs humains, nous permettant d'obtenir de meilleures performances d'alignement automatique, pour *in fine* simplifier la tâche d'annotation manuelle via de meilleurs pré-alignements.

Alignements de référence Les modèles d'alignements automatiques sont évalués par rapport à des alignements issues de l'annotation humaine, détaillé en section 4.2.

Exploration des paramètres Afin d'évaluer le grand nombre de combinaisons possibles de paramètres à un coût raisonnable en termes de calcul, nous utilisons plusieurs jeux de données de validation de tailles différentes. Nous définissons, pour chaque paramètre, un ensemble de valeurs possibles, et calculons les résultats pour chaque combinaison (*Grid Search*). Le nombre de combinaisons évolue alors proportionnellement au produit des cardinaux des ensemble de chaque paramètres. Afin de limiter le coût en calculs, nous exécutons les expériences itérativement, d'abord avec toutes les combinaisons de paramètres sur un petit jeu de données, puis, en sélectionnant un sous-ensemble des meilleures combinaisons sur un jeu de données plus grand et ainsi de suite. Ceci nous permet d'explorer l'espace des paramètres sans donner trop d'importances aux combinaisons trop peu performantes tout en identifiant les paramètres les plus importants.

1ère itération : Alignement diagonal Lors de notre première itération nous ne disposons d'aucune donnée de référence (*i.e.* données annotées). Par conséquent, nous n'avons à ce stade aucun moyen d'évaluer quantitativement la performance de l'alignement automatique. Afin de fournir un pré-alignement aux annotateurs humains, nous proposons un alignement naïf consistant à aligner les segments en diagonale. On considère par exemple que la phrase située à la moitié de la transcription est alignée avec la phrase à la moitié du compte rendu. Ainsi, ni les similarités S ni l'alignement A ne sont pris en compte, l'alignement se contente de suivre la diagonale reliant les coordonnées $(1, 1)$ à $(|\boldsymbol{\mathcal{T}}|; |\boldsymbol{\mathcal{R}}|)$. L'historique H de l'eq. 4.9 est modifié comme suit (eq. 4.11) :

$$
\begin{aligned}
H_{i,j} &= \begin{cases} (i-1,j) \text{ si } r_{i,j} < r \\ (i,j-1) \text{ sinon} \end{cases} \\
\text{avec } r &= |T|/|R| \\
\text{et } r_{i,j} &= (i-1)/(j-1)
\end{aligned}
\tag{4.11}
$$

2$^{\text{nde}}$ Itération : exploration des fonctions de similarité Durant la seconde itération, nous avons principalement exploré les différentes représentations du texte et les fonctions de similarité (voir 4.3.1). Nous avons comparé le score ROUGE [Lin, 2004], plus précisément les variantes R1-F, R2-F, and RL-F.

Comme indiqué précédemment, nous avons étudié l'utilisation de deux représentations vectorielles : $tf \cdot idf$ avec *LSA* (*Latent Semantique Analysis*, Daumé et Marcu, 2009), ainsi que des plongements de mots pré-entrainés par [Fauconnier, 2015] avec `word2vec` [Mikolov *et al.*, 2013c]. La similarité cosinus est utilisée comme fonction de score entre vecteurs.

Mesurer la similarité au niveau des fenêtres glissantes au lieu de travailler au niveau des phrases directement vise à limiter l'impact de phrases courtes, isolées ou du bruit sur les similarités. En effet, les données, en particulier du côté de la transcription, pouvant être bruitées, certaines phrases peuvent obtenir une similarité très faible au sein même de segments traitant du même sujet. Les paramètres relatifs aux fenêtres glissantes sont la taille (s, pour *size*, expriméé en nombre de mots) des fenêtres et le chevauchement (o, pour *overlap*). Nos expériences explorent les combinaisons de ces paramètres tels que $s \in \{0, 1, 2, 3, 4, 5, 10\}$ et $o \in \{1, 2, 3, 5\}$.

Les représentations des fenêtres sont obtenues par *agrégation*(agg) des représentations des phrases. Enfin, les scores des phrases sont calculés par *réduction* (red) des scores des fenêtres. Nous expérimentons avec $agg \in \{sum, mean, max\}$, $red \in \{sum; product\}$.

3$^{\text{ème}}$ Itération : réglage fin (*fine tuning*) des modèles basés sur les plongements de mots Pendant l'alignement, nous avons observé que l'algorithme de programmation dynamique à la base de l'alignement automatique pouvait garder la même direction sur un grand nombre de phrases. Par exemple, un segment du compte rendu peut obtenir une grande similarité avec un grand nombre de phrases successives de la transcription, et donc sembler aligné avec toutes ses phrases, ayant pour résultat un alignement trop monotone. Pour limiter ce comportement, nous introduisons des facteurs de pénalités

(typiquement dans $[0;1]$) afin de « décourager » les mouvements monotones horizontaux (*hd*, pour *horizontal decay*) et verticaux (*vd*, pour *vertical decay*). En résulte la matrice d'alignement pénalisée A' telle que :

$$\begin{aligned} A'_{i,j} &= A_{i,j} \times D_{i,j} \\ D_{i,j} &= \begin{cases} D_{i-1,j} \times (1 - hd) \text{ si } A_{i-1,j} > A_{i,j-1} \\ D_{i,j-1} \times (1 - vd) \text{ sinon} \end{cases} \end{aligned} \tag{4.12}$$

La pénalité est réinitialisée, $D_{i,j} = 1$ à chaque changement de direction. L'historique H et le chemin optimal P sont alors calculés de la même manière que précédemment, mais à partir de A'.

4ème Iteration : `public_meetings` Enfin, nous sélectionnons un ensemble de réunions à caractère public afin de pouvoir le diffuser à des fins d'expérimentations et de réplication des résultats. Ce jeu de données plus petit est utilisée comme ensemble de test pour nos expériences et est donc exclu des données d'apprentissage et de développement.

4.3.4 Évaluation

Systèmes de base

La tâche d'alignement telle que nous l'abordons est proche de celle de la segmentation linéaire, qui consiste à placer des frontières thématiques, c'est-à-dire à isoler les portions de texte qui traitent de sujets différents. En cela, nous pouvons aborder l'alignement en tant que segmentation linéaire de la transcription, en associant chronologiquement les segments transcriptions aux comptes rendus (à condition de produire le même nombre de segments). Nous proposons alors de comparer nos modèles d'alignement par rapport aux modèles de segmentation linéaire. En particulier, nous considérons dans nos expériences plusieurs modèles de référence issus du domaine de la segmentation linéaire, à savoir : TextTiling de [Hearst, 1997] et C99[Choi, 2000].

Toutefois, ces modèles de segmentation linéaire sont pénalisés par rapport à nos méthodes d'alignement du fait que la segmentation ne tire pas profit de la connaissance des comptes rendus. Nos modèles n'ont pas besoin par exemple de prédire le nombre de segments à trouver, ce nombre étant fixé par la segmentation du compte rendu. Par conséquent, afin de rendre la comparaison plus juste, nous ne considérons, pour les modèles de segmentation, que les ensembles de paramètres qui produisent le nombre de segments at-

tendus. Ce nombre peut être passé directement dans C99 alors que nous devons chercher les bons paramètres, par *grid search* dans le cas TextTiling.

La segmentation linéaire est évaluée avec la métrique `WindowDiff` [Pevzner & Hearst, 2002], qui compare les bornes des segments prédites par rapport à la référence dans une fenêtre glissante de taille k. `WindowDiff` a été proposée comme alternative à P_k [Beeferman *et al.*, 1999] avec l'objectif d'être plus juste par rapport aux faux négatifs, au nombre de bornes, à la taille des segments et aux erreurs proches. Nous évaluons donc les modèles de segmentation linéaire ainsi que nos expériences d'alignement avec le score `WindowDiff`. Nous reportons également des métriques plus simples, telles que l'exactitude au niveau des segments (nombre de segments bien segmenté), et l'exactitude au niveau des mots (nombre de mots bien segmentés)

Évaluation de l'alignement en tant que méthode d'augmentation de la taille des jeux de données pour le résumé automatique

La motivation de nos travaux sur l'alignement automatique est d'annoter automatiquement nos jeux de données afin d'améliorer les performances des modèles de résumé automatique. Par conséquent, nous comparons la performance de modèles sans alignement automatique, c'est-à-dire entraînés uniquement sur des données annotées manuellement, par rapport aux modèles entraînés avec les données alignées manuellement et automatiquement. Le traitement des données inclut des filtres sur le nombre de mots ($\#mots$) et de phrases ($\#phrases$) présents dans les segments. Ne sont considérés que les segments tels que : $10 \leq \#mots \leq 1000$ et $3 \leq \#phrases \leq 50$.

Grâce à OpenNMT-py [2][Klein *et al.*, 2017] nous entraînons des modèles Transformer [Vaswani *et al.*, 2017] similaires aux baseline de [Z. M. Ziegler *et al.*, 2019], à la différence que nous désactivons le mécanisme de copie (sec. 3.2.3).

De plus, après avoir mesuré les longueurs des paires d'entraînement (en nombre de mots et de phrases), nous constatons une forte variance au niveau de la longueur des résumés et du taux de compression (rapport de réduction de longueur du résumé par rapport à la source). Ces mesures sont présentées et comparées aux jeux de données de la littérature dans le tableau 1.2. En particulier, on observe que le coefficient de variation [3] (c_v) mesurés sur la longueur des résumés est de 1.1 (contre 0.4 pour CNN/DM) et de 2.15

2. https://github.com/OpenNMT/OpenNMT-py

3. Le coefficient de variation $c_v = \sigma/\mu$ est une mesure de dispersion sans unité. Ce coefficient exprime le degré de variabilité par rapport à la moyenne.

sur le taux de compression (contre 0.06 pour CNN/DM). Nous supposons alors que cette variance peut poser un problème aux modèles de résumé et nous proposons d'ajouter dans la source une information quant à la longueur du résumé souhaitée. En pratique, nous représentons cette information par 10 tokens spéciaux (de `LEN_0` à `LEN_9`) représentant des classes de longueurs. Ces classes correspondent aux déciles mesurés sur longueurs (en nombre de mots) des résumés cibles au sein des données alignées manuellement et automatiquement). Par exemple, le token `LEN_0` informe que l'on souhaites un résumé qui soit dans les 10% les plus courts par rapport aux jeux de données d'entraînement. Cette approche est motivée par l'introduction par [Fan *et al.*, 2018] de tokens spéciaux pour contrôler des paramètres de la génération – la longueur dans notre cas. Nous proposons d'approfondir cette approche au chapitre 6. Dans nos évaluations, nous comparons les performances des modèles *avec* et *sans* l'information de la longueur du résumé souhaité, afin de mesurer l'impact d'une telle information pour la génération de résumés.

Les évaluations sont menées sur l'ensemble de test `public_meetings` et utilisent la métrique ROUGE(F) (Lin, 2004, *cf.* sec. 1.3.1).

4.4 Résultats

4.4.1 Alignement Automatique

Les performances des modèles d'alignement automatique sont présentées dans le tableau 4.1.

Les résultats sont exprimés en termes d'exactitude au niveau des segments (*seg*%, eq. 4.13) ; exactitude au niveau des mots (*word*%, eq. 4.14) ; et avec WindowDiff (métrique de segmentation linéaire, [Pevzner & Hearst, 2002])

$$seg\% = \frac{\#segments\_correctement\_alignés}{\#segments\_total} \tag{4.13}$$

$$word\% = \frac{\#mots\_correctement\_alignés}{\#mots\_total} \tag{4.14}$$

Les modèles de segmentation linéaire d'une part, et les approches d'alignement basées sur TF-IDF et le score ROUGE d'autre part ne semblent pas être des directions prometteuses étant donné qu'elles ne parviennent pas à battre la référence naïve : l'alignement diagonal.

Model	**Window** (s, o)	**Window Scoring** (agg, red)	**Alignment** (hd, vd, p)	**Dev. Acc.** $(seg.\%, word\%)$	**Dev. WD** ↓	**Test Acc.** $(seg.\%, word\%)$	**Test WD** ↓
TextTiling	–	–	–	–	–	09.36 – 06.66	39.61
C99	–	–	–	–	–	05.68 – 05.03	42.49
Diagonal	–	–	–	18.59 – 21.55	17.43	20.75 – 23.28	34.61
tf-idf	2 – 1	sum – sum	$0\,;\,0\,;\,4$	5.02 – 5.23	13.80	4.10 – 5.17	23.02
	10 – 3	mean – mul	$10^{-4}\,;\,10^{-4}\,;\,1$	9.69 – 10.90	17.33	9.91 – 10.94	33.00
	1 – 0	max – prod	$10^{-4}\,;\,10^{-4}\,;\,2$	10.93 – 12.33	17.74	10.29 – 10.90	35.05
ROUGE	10 – 2	cat – sum	$10^{-4}\,;\,10^{-4}\,;\,1$	11.43 – 12.70	17.157	9.25 – 9.66	32.99
	2 – 0	cat – sum	$10^{-4}\,;\,10^{-4}\,;\,4$	14.52 – 16.95	17.85	10.63 – 11.76	34.01
Embeddings	2 – 1	sum – prod	$0\,;\,0\,;\,4$	45.62 – 54.06	10.81	64.36 –72.95	19.872
	2 – 1	sum – prod	$10^{-4}\,;\,10^{-4}\,;\,4$	60.96 – 70.73	10.94	58.97 – 68.09	23.43
	2 – 1	sum – prod	$0\,;\,10^{-4}\,;\,4$	**61.00 – 72.50**	**10.38**	**69.36 – 79.06**	**15.09**

TABLE 4.1 – Évaluation des modèles d'alignement automatique par rapport aux données de validation (202 réunions de référence) et l'ensemble de test `public_meetings` (22 réunions) sur trois métriques : exactitude-segments ($seg\%$), exactitude-mots ($word\%$) et WindowDiff [PEVZNER & HEARST, 2002].

En revanche, les approches basées sur les plongements de mots se distinguent très nettement, avec des performances deux fois supérieures à l'alignement diagonal (en termes d'exactitude) et trois fois supérieures aux autres modèles sur l'ensemble de validation.

L'introduction de pénalités lors de l'alignement (voir eq. 4.12), avec pour but de réduire la monotonie de l'alignement s'est faite progressivement, en partant de très faibles valeurs que ce soit pour la pénalité horizontale ou verticale. Les résultats font ressortir la pertinence de ces pénalités. La pénalité verticale (vd) est particulièrement importante avec un écart de près de 15% et 18% en termes d'exactitude au niveau segment et mot respectivement. La pénalité horizontale semble avoir un impact bien inférieur, et dégrade même légèrement les meilleurs modèles.

De même, on observe que le fait d'élever les scores à la puissance $p > 1$ (voir eq. 4.8) durant l'alignement améliore les performances de chaque modèle. Ceci peut en effet permettre de discriminer plus nettement les scores.

Les fenêtres glissantes apportent également un gain de performances par rapport aux représentations de phrases (*i.e.* $s = 1, o = 0$) dans la plupart des cas – exception faite des modèles TF-IDF. Nous observons différentes configurations pour les différents modèles, tant en termes de taille de la fenêtre s et de *l'overlap* o ou de fonction d'agrégation (agg) et de réduction (red). Les modèles avec plongements de mots quant à eux semblent converger vers des fenêtres de 2 mots avec un *overlap* de 1 en utilisant la somme comme fonction d'agrégation des représentations et le produit comme fonction de réduction des scores.

4.4.2 Évaluation Humaine

Dans le cadre de l'alignement, l'annotation humaine consiste à corriger le pré-alignement fourni par les modèles d'alignement automatique (*cf.* 4.2). À mesure de l'amélioration des performances d'alignement automatique, nous avons pu fournir de meilleurs pré-alignements, afin de faciliter et d'accélérer l'annotation. Ainsi, nous pouvons mesurer la qualité du pré-alignement grâce aux nombres de corrections apportées par l'annotateur. Nous calculons alors l'exactitude du modèle d'alignement comme étant le ratio de segments que l'annotateur a considéré bon, c'est-à-dire qu'il n'a pas modifié.

Les résultats de l'évaluation humaine sont présentés dans le tableau 4.4), et figure 4.2. On observe une augmentation nette du score d'une itération à l'autre.

	#documents	**% Segments Corrects** ↑ $(mean, median)$
1rst iteration	12	18.63 – 15.73
2rst iteration	88	50.44 – 53.56
3rd iteration	38	57.23 – 55.02
`public_meetings`	22	72.67 – 80.08

TABLE 4.2 – Évaluation humaine de l'alignement automatique.

4.4.3 L'alignement automatique au service du résumé

L'objectif de l'alignement automatique est de fournir davantage de données pour l'entraînement de modèles de résumé automatique. Nous évaluons alors si cet ajout de données est bénéfique aux modèles en entraînant d'abord un modèle uniquement sur les données annotées (20,000 paires d'entraînement) ; puis un autre modèle à partir des données alignées manuellement et automatiquement (70,000 paires d'entraînement supplémentaires). Nous observons un gain substantiel de performance grâce à l'alignement automatique, qui se traduit par une hausse de +7 points de score ROUGE sur les prédictions *sans information* (table 4.3). Par ailleurs, l'ajout de l'*information de longueur* hisse les performances des modèles à un tout autre niveau en ajoutant +8.77 R1 à l'entraînement sur les données manuelle (`man`) et +5.34 au modèle `man + auto`.

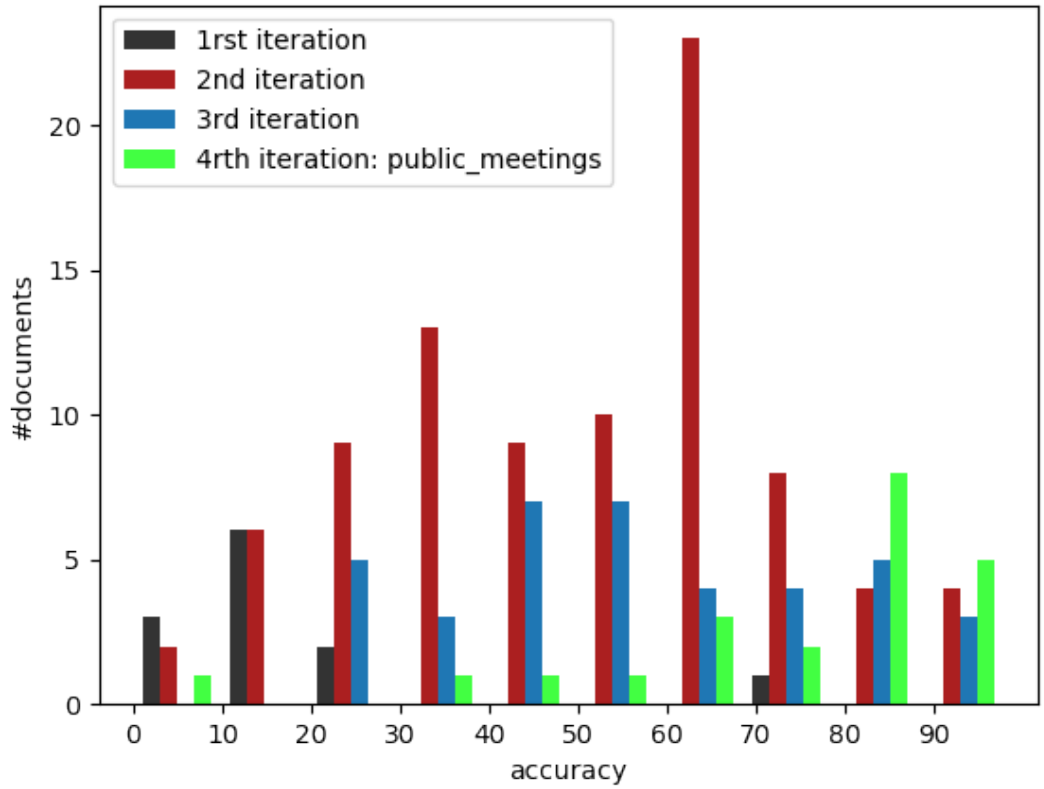


FIGURE 4.4 – Nombre de documents alignés par itération et leur répartition par exactitude moyenne des pré-alignements automatique par rapport aux corrections des évaluateurs humains.

Jeu de données	**# Paires** $(train, test)$	**ROUGE-F** (R1 / R2 / RL) *sans information*	*information longueur*
`man`	21k – 1060	44.03 / 23.74 / 41.09	52.80 / 29.59 / 49.49
`man` + `auto`	91k – 1060	51.22 / 32.41 / 48.65	56.56 / 35.43 / 53.55

TABLE 4.3 – Scores ROUGE évalués sur l'ensemble de test `public_meetings` des modèles de résumé automatique entraînés, d'une part, sur les données de références (`man` : annotée manuellement) ; et d'autre part en ajoutant les données alignées automatiquement (`auto`). Le score ROUGE est mesuré d'abord *sans information*, puis *avec information* de la longueur du résumé souhaité (un token spécial par décile).

4.5 Conclusion

Dans ce chapitre, nous avons présenté la création d'un jeu de données pour l'apprentissage de la génération automatique de comptes rendus de réunions. Nous proposons une méthodologie s'appuyant sur l'annotation humaine et le développement de méthodes d'alignement automatique. Nous avons procédé sous forme d'itérations comprenant (i) la génération de pré-alignements automatiques ; (ii) l'annotation humaine de l'alignement sous forme de correction du pré-alignement ; (iii) le développement et l'évaluation de méthodes d'alignement automatique. Ces itérations nous permettent d'une part de mieux évaluer nos méthodes d'alignement automatique grâce à une quantité croissante de données de référence, et d'autre part de faciliter le travail d'annotation grâce à l'amélioration de la qualité des pré-alignements.

Trois approches d'évaluation sont ensuite proposées : en comparant nos modèles d'alignement automatique avec des modèles de segmentation linéaire ; en comparant les modèles sur la base du nombre de corrections apportées par les annotateurs (*i.e.* moins il y a de corrections à effectuer plus on considère que le modèle est de qualité) ; et enfin en étudiant l'impact des données alignées automatiquement sur la qualité des modèles de résumé automatique.

Nous obtenons alors que nos modèles d'alignements sont largement plus performants que les modèles de segmentation linéaire. Ce résultat n'est pas très surprenant dans la mesure où la comparaison n'est pas égale : l'alignement s'effectue avec la connaissance du compte rendu, alors que la segmentation linéaire n'utilise que la transcription. Les modèles utilisant les *word embeddings* présentent des résultats significativement meilleurs que les autres approches (TF-IDF, ROUGE), que ce soit en termes d'exactitude de l'alignement ou du score WindowDiff (4.1). Ces résultats se traduisent également par l'amélioration du jugement humain itération après itération (*cf.* 4.2).

Enfin, nous confirmons expérimentalement la pertinence de l'alignement automatique dans le cadre de la constitution d'un jeu de données. En effet, l'entraînement de modèles de résumé automatique est nettement plus performant lorsque l'on inclut les données alignées automatiquement. Sans fournir d'information supplémentaire sur la longueur des résumés cibles, on observe une hausse supérieure à +7 points ROUGE (R1, R2, RL) par rapport au modèle entraîné uniquement sur les données alignées manuellement. En passant l'*information de longueur*, le modèle avec alignement manuel et automatique conserve un net avantage de +4 points ROUGE. Enfin, la différence de performance liée à l'information

de longueur est respectivement de +8.77 R1 et +5.34 R1 pour les modèles entraînés sur `man` et `man+auto`. Ceci confirme notre hypothèse que la longueur du résumé cible est un aspect très important de la génération, un modèle parvenant à prédire cette information précisément aurait donc un avantage non négligeable.

À l'issue de ces travaux d'alignement, nous publions le jeu de données `public_meetings`, aligné manuellement, que nous avons utilisé comme ensemble de test.

Chapitre 5

Apprentissage à partir de données non alignées

L'alignement des données, abordé en chapitre 4, constitue un frein à la collecte de données. L'alignement humain est coûteux et long, et l'alignement automatique est limité par le coût en calcul de la transcription automatique ainsi que part la nécessité de disposer de paires de fichiers correspondant à l'audio de la réunion et à son compte rendu.

Par conséquent, nous explorons dans ce chapitre des alternatives à l'alignement des données en cherchant à tirer profit des données non alignées. Nous proposons deux approches : (i) l'*apprentissage auto-supervisé* permettant de pré-entraîner les modèles à partir de tous les comptes rendus dont dispose Ubiqus ; (ii) nous introduisons la notion de *résumé inverse* pour la génération de transcriptions synthétiques correspondant aux comptes rendus non alignés, permettant d'obtenir des paires *(transcription synthétique ; compte rendu)*.

Table des matières

5.1 Problématique

Durant le chapitre 4, nous avons exploré l'alignement des transcriptions avec les comptes rendus de réunion, grâce à des annotateurs humains (4.2) et avec l'introduction de modèles d'alignement automatique (4.3). Nous avons ainsi pu obtenir $90,000$ paires d'entraînement, dont $70,000$ grâce à l'alignement automatique. Ce résultat est encourageant, et nous a déjà permis d'améliorer les performances des modèles de résumé automatique (*cf.* 4.3). Toutefois, l'alignement – même automatique – nécessite du contrôle et des opérations humaines. En effet, les fichiers correspondant aux paires *(transcription, compte rendu)* ne respectent pas nécessairement de convention de nommage, il est donc difficile d'automatiser totalement le processus de recherche de données. De plus, l'alignement automatique s'appuie sur la transcription automatique des enregistrements audio des réunions. Cette transcription automatique est coûteuse en calcul.

Or, les données alignées (manuellement et automatiquement) constituent une faible proportion du total des comptes rendus D'UBIQUS : on dispose de $9 \cdot 10^4$ segments alignés contre un total plus de $5 \cdot 10^6$ comptes rendus UBIQUS disponibles.

Nous proposons donc deux approches afin de tirer profit de cette grande quantité de comptes rendus non alignés. La première est basée sur l'apprentissage auto-supervisé permettant de pré-entraîner nos modèles sur des données non alignées. Pour la seconde approche, nous introduisons le *résumé inverse*, c'est-à-dire la tâche consistant à générer la transcription à partir du compte rendu (et non le compte rendu à partir de la transcription). Nous entraînons des modèles de résumé inverse et les utilisons afin de générer des transcriptions synthétiques pour chaque compte rendu non aligné et ainsi créer un nouveau jeu de données.

L'approche par apprentissage auto-supervisé est présentée en section section 5.2 ; l'approche par résumé inverse en section 5.3. Nous détaillons les expériences et les résultats en section 5.4, avant de discuter les résultats, et de conclure (sec. 5.5).

5.2 Valorisation des données non-alignées grâce à l'apprentissage auto-supervisé

Les modèles de résumé automatique que nous avons entraînés jusque-là reposent sur un entraînement *supervisé*, c'est-à-dire que nous fournissons des paires d'entraînement composées de *l'entrée* et de la *sortie* souhaitée (aussi appelée la *cible*). Cette approche

nécessite, de fait, des données alignées.

À l'inverse, une approche *auto-supervisée* vise à apprendre à partir des *entrées* seules. On peut par exemple entraîner un modèle à reconstruire une séquence de mots à partir d'une sous-séquence, c'est-à-dire à prédire les mots manquants ou les mots suivants. C'est en particulier le cas des modèles de langage (sec. 2.2.2) où l'on cherche à prédire des mots à partir de leur contexte (les mots précédents, ou les mots voisins).

Inspiré par les récents succès des modèles pré-entraînés par apprentissage auto-supervisé (voir 2.5) montrant d'impressionnantes capacités de transfert d'apprentissage, nous proposons de valoriser les données non alignées dont dispose UBIQUS grâce à un pré-entraînement auto-supervisé. Suivant l'approche proposée par BART [LEWIS *et al.*, 2020], le pré-entraînement consiste, pour le modèle, à reconstruire un texte correct à partir d'une version bruitée de celui-ci. [LEWIS *et al.*, 2020] proposent plusieurs fonctions de bruit :

1. **Masquage :** certaines portions du texte sont remplacées par un *seul* mot spécial `MASK`. La longueur de chaque portion de texte suit une loi de Poisson de paramètre λ_{mask}. Le paramètre p_{mask} correspond à la proportion du texte à masquer. Par exemple, si $p_{mask} = 0.3$, alors 30% des mots seront masqués, repartis dans plusieurs portions de texte.
2. **Permutation de phrases :** les phrases sont permutées aléatoirement avec une probabilité p_{shuf}.
3. **Suppression de mots :** on supprime des mots avec une probabilité p_{del}.
4. **Rotation du document :** un mot est choisi aléatoirement (uniformément) pour être le premier mot. Les mots précédents sont déplacés en fin de séquence.

En pratique, [LEWIS *et al.*, 2020] obtiennent leurs meilleures performances sur la tâche du résumé automatique en utilisant uniquement le masquage (avec $lambda_{mask} = 3$, $p_{mask} = 0.3$) et la permutation des phrases (avec $p_{shuf} = 1$). L'architecture de BART repose sur un TRANSFORMER classique, comportant encodeur et décodeur, par opposition à d'autres modèles qui ne pré-entraînent que l'encodeur (*e.g.* BERT DEVLIN *et al.*, 2019) ou le décodeur (*e.g.* GPT RADFORD *et al.*, 2018 ; RADFORD *et al.*, 2019 ; BROWN *et al.*, 2020). Par conséquent, l'implémentation, l'entraînement et le *finetuning* d'un tel modèle est particulièrement simple : il suffit d'implémenter et d'appliquer les fonctions de bruit à la source. La phase de *finetuning* consiste alors à reprendre l'entraînement de manière supervisée avec cette fois les données alignées : avec les *transcriptions* en entrées et les *comptes rendus* en sorties souhaitées. Nous présentons cette approche avec pré-entraînement auto-supervisé suivis du *finetuning* supervisé dans la figure 5.1.

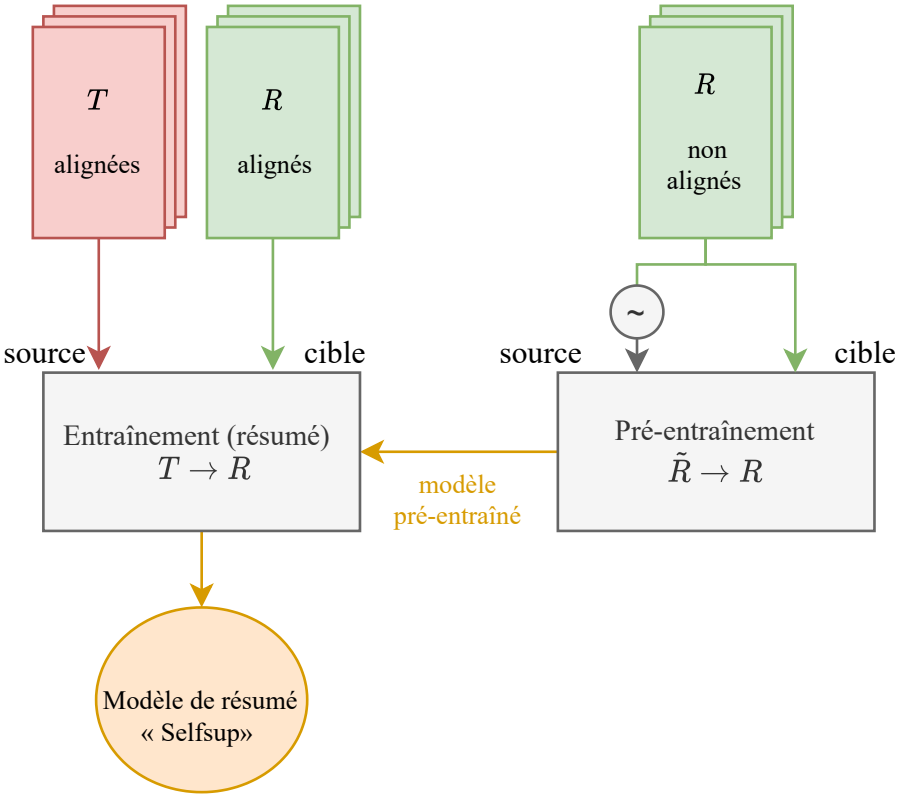


FIGURE 5.1 – Approche auto-supervisée : on pré-entraîne un modèle à reconstruire les comptes rendus non alignés (*R non alignés*), à partir d'une version bruitée par la fonction $\sim$ de ceux-ci. Ensuite, le *finetuning* (à gauche) consiste à ré-entraîner le modèle sur la tâche de génération de comptes rendus (*R alignés*) à partir de transcriptions (*T alignées*).

5.3 Génération et utilisation de données synthétiques grâce au résumé inverse

Une autre manière de tirer profit de données non-alignées est de générer les données manquantes. Bien qu'à notre connaissance ce ne soit pas une pratique courante dans le domaine du résumé automatique, cela a été largement étudié dans le cadre de la traduction automatique sous le terme de *traduction inverse* (*back-translation* Sennrich *et al.*, 2016 ; Edunov *et al.*, 2018 ; Poncelas *et al.*, 2018). De manière similaire, nous introduisons le *résumé inverse* (*back-summarization*) consistant à générer le texte source à partir du résumé, c'est-à-dire dans notre cas de produire des transcriptions synthétiques à partir des comptes rendus non-alignés.

Le *résumé inverse* consiste en trois étapes (représentées en figure 5.2) :

1. **Entraînement inverse** : en utilisant les données alignées manuellement (`man`) et automatiquement (`auto`), nous entraînons un *modèle inverse* à prédire les *transcriptions* à partir du *compte rendu*.
2. **Génération des transcriptions synthétiques** : à partir des comptes rendus non-alignés, nous générons les *transcriptions* synthétiques correspondantes grâce au *modèle inverse*. Nous obtenons ainsi un nouveau jeu de données, que nous notons `back`.
3. **Entraînement du résumé** : à partir des trois jeux de données alignés (`man + auto + back`), nous entraînons un modèle de résumé automatique « classique » à générer les *comptes rendus* à partir des *transcriptions*.

Les modèles de résumé et de résumé inverse sont similaires aux modèles de référence en termes d'architecture et d'hyper-paramètres (voir sec. 5.4). Les transcriptions synthétiques sont générées à partir du meilleur modèle inverse en termes de score ROUGE sur l'ensemble de validation. Nous cherchons aussi à diminuer *l'extractivité* du modèle, c'est-à-dire sa tendance à copier depuis la source (ce point est discuté plus en détail en 5.4.2). L'inférence utilise la recherche en faisceau (*beam search*) de taille 5 et bloque les répétitions de trigrammes [Paulus *et al.*, 2017].

Durant l'apprentissage, les différents jeux de données (`man`, `auto` et `back`) sont pondérés. Ces poids déterminent les proportions de paires d'entraînement tirées dans chaque jeu de données. Par exemple, des poids de $(1, 10, 100)$ pour respectivement `man`, `auto`, et `back`, signifient que pour chaque paire d'entraînement tirée dans `man` le modèle va itérer sur 10

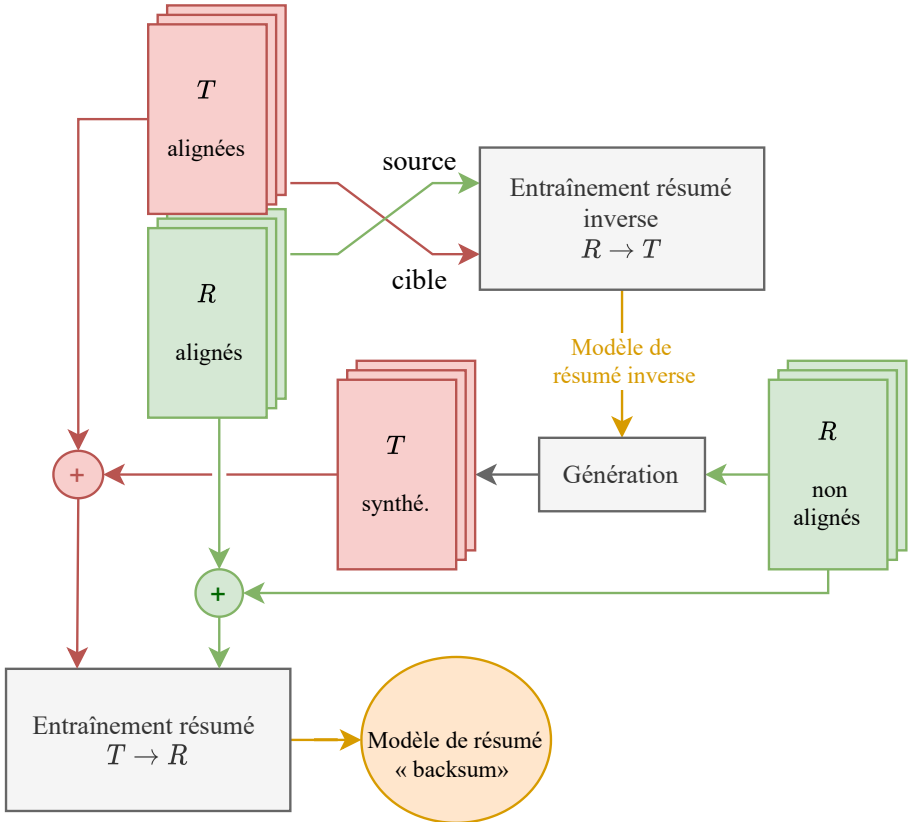


FIGURE 5.2 – Les trois étapes du *résumé inverse*. (1) On entraîne d'abord un modèle sur la tâche inverse, c'est-à-dire à prédire les transcriptions (T *alignées*) à partir des comptes rendus (R *alignés*). (2) Le modèle inverse est ensuite utilisé pour générer les transcriptions synthétiques (T *synthétique*) à partir des comptes rendus non alignés (R *non-alignés*). (3) Ces paires *(transcription synthétique ; compte rendu non aligné)* ainsi obtenues sont finalement utilisées en plus des données initiales (T *alignées* ; R *alignés*) pour l'entraînement du modèle de résumé.

paires issues de `auto` et 100 de `back`. Cette pondération est motivée par [Edunov *et al.*, 2018] qui suggèrent (section 5.5) le *suréchantillonnage* (*upsampling*) c'est-à-dire d'utiliser plus fréquemment les données annotées par rapport aux données synthétiques. Afin de rester comparable aux autres modèles, nous ne changeons pas la pondération entre `man` et `auto` $(2, 7)$ qui correspond à leur taille respective $(20,000$ et $70,000$ segments). Pour les données synthétiques, nous utilisons un poids de 100, ce qui correspond à un taux de suréchantillonnage de 6, comme suggéré dans [Edunov *et al.*, 2018].

5.4 Expériences

5.4.1 Modèle de référence et cadre expérimental

Nous partons des expériences menées dans le cadre de l'alignement automatique (sec. 4.3.2, [Tardy *et al.*, 2020a]) et prenons comme référence le meilleur modèle, c'est-à-dire celui entraîné sur les données alignées manuellement (`man`) et automatiquement (`auto`). Les modèles sont des Transformer [Vaswani *et al.*, 2017] standard, de 6 couches pour l'encodeur et pour le décodeur, 8 têtes d'attention et des plongements de mots à 512 dimensions. Les hyper-paramètres du modèle et de l'optimisation sont similaires à ceux du modèle *baseline*) de [Z. M. Ziegler *et al.*, 2019]. Comme seuls changements majeurs, nous désactivons le mécanisme de copie, ainsi que le partage des poids des plongements de mots, ainsi que discuté en 5.4.2).

Dans les sections suivantes, nous étudions comment des modèles simples – utilisant tous la même architecture et les mêmes hyper-paramètres que précédemment – peuvent tirer profit d'une grande quantité de comptes rendus non-alignés.

5.4.2 Réduction de la copie

Durant l'entraînement et la validation des modèles de résumé inverse, nous avons rencontré le problème du *biais d'extractivité*. C'est une observation commune dans le domaine du résumé automatique que les modèles tendent à devenir trop extractifs, c'est-à-dire à trop copier le texte d'entrée (ce biais est développé en section 3.6). Ceci est vrai depuis les modèles basés sur des *RNN/LSTM* [See *et al.*, 2017] mais aussi pour les plus récents – et plus « abstractifs » – comme Bart [Lewis *et al.*, 2020].

Notre approche basée sur le résumé inverse amplifie ce biais du fait qu'elle implique deux modèles – un de résumé inverse, l'autre de résumé – qui sont tous deux biaisés. Les

premiers entraînements de modèles TRANSFORMER pour le résumé ont été effectués avec mécanisme de copie [SEE *et al.*, 2017] et partage des poids entre (i) plongements de mots de l'encodeur et du décodeur ; (ii) les plongements de mots et le générateur, de manière similaire à [Z. M. ZIEGLER *et al.*, 2019]. Lors de l'évaluation par rapport à l'ensemble de validation – qui présente un taux de copie de 55.38% –, les prédictions contenaient trop de copie : 62.64%, soit +7.26% (en absolu).

Nous avons observé que désactiver le mécanisme de copie et le partage des poids entre le décodeur et le générateur [1] permettait d'abaisser le taux de copie des prédictions à 54.65, (−0.73). De plus, l'ensemble de données ainsi obtenu présente un taux de copie moyen de 60.37 (au lieu de 76.31% avant réduction) ce qui est proche des autres ensembles de données d'UBIQUS (voir tableau 5.1).

5.4.3 Jeux de données

Les jeux de données utilisés pour les expériences suivantes sont présentés dans le tableau 5.1. On note respectivement `man` et `auto` les jeux de données alignés manuellement et automatiquement. Le jeu de données `back` est composé des comptes rendus non-alignés associés aux transcriptions synthétiques obtenues par *résumé inverse* (voir 5.3).

L'ensemble de validation – `valid` – est échantillonné à partir des données alignées manuellement. Enfin, nous utilisons `public_meetings` [TARDY *et al.*, 2020a] comme ensemble de test. Nous observons des différences conséquentes entre `public_meetings` et `valid` en termes de longueur de séquence et de taux de copie. Nous supposons que les contraintes spécifiques de ce corpus – à savoir ne comporter que des réunions publiques – ont introduit un biais. Par conséquent, nous conduisons nos évaluations sur `public_meetings` et `valid`.

5.4.4 Résumé automatique

Les résultats des modèles de résumé automatique sont présentés dans les tableaux 5.2 pour l'ensemble de test (`public_meetings`) et 5.3 pour l'ensemble de validation (`valid`).

Les performances de l'approche avec pré-entraînement auto-supervisé (SELFSUP) surpassent nettement celles du modèle de référence, avec un écart de +2.7 ROUGE-1 et ROUGE-2 sur l'ensemble de test et jusqu'à +4 sur l'ensemble de validation en termes de ROUGE-1 and ROUGE-2.

1. Il est courant que les modèles de résumé automatique partagent les poids des couches de plongement de mot du décodeur avec la couche de projection du générateur.

TABLE 5.1 – Jeux de données dans le cadre de l'évaluation de l'augmentation des données. Sont reportés le nombre de paires d'entraînement ; la longueur des séquence d'entrées (src) et cibles (tgt) en nombre de mots moyen, premier décile d_1 et dernier d_9) ; et le taux de copie mesuré en termes de ROUGE-1 entre l'entrée et la sortie de référence.

Dataset	**#Pairs**	**src** $(avg, [d_1, d_9])$	**tgt** $(avg, [d_1, d_9])$	**Copy %** $R1$
`man`	21k	172, [42, 381]	129, [25, 297]	55.45
`auto`	68k	188, [45, 421]	130, [25, 302]	54.72
`valid`	1k	178, [40, 409]	144, [22, 294]	55.38
`back`	6.3m	232, [53, 509]	90, [43, 153]	60.37
`public_meetings`	1060	261, [52, 599]	198, [33, 470]	75.84

De même, le modèle BACKSUM, utilisant les données synthétiques issues du résumé inverse surclasse largement le modèle de référence : +2.98 R1, +2.39 R2 sur l'ensemble de test, et +5.34 R1, +4.80 R2 sur l'ensemble de validation.

Nous entraînons enfin un troisième modèle, BOTH, combinant à la fois un pré-entraînement auto-supervisé et un *finetuning* avec données synthétiques. Il est intéressant de noter que ce modèle, qui est en quelque sorte la combinaison des deux autres, est le meilleur des trois, et dépasse très nettement la référence sur l'ensemble de test (+5.71 R1 et +4.64 R2) et de validation (+6.40 R1 et +5.16 R2).

En plus d'observer le score ROUGE nous prêtons attention au taux de copie des modèles. Comme discuté en section 5.4.2, nous remarquons que les modèles sont biaisés en faveur d'un résumé extractif, du fait qu'ils génèrent des prédictions avec de forts taux de copie sur les deux ensembles d'évaluation (respectivement 75.85 copy% et 55.38 copy% pour `test` et `valid`). Bien que nous ayons veillé à limiter ce biais lors du résumé inverse (*cf.* 5.4.2) nous observons que les modèles utilisant le résumé inverse (donc BACKSUM et BOTH) ont un taux de copie supérieur au modèle auto-supervisé (SELFSUP).

5.5 Discussions et Conclusion

Taux de copie sur les différents jeux de données Conduire l'évaluation sur deux jeux de données différents a permis de comparer comment les modèles se comportent lorsque confrontés à des entrées de natures différentes. En particulier, on observe une différence de 20% sur le taux de copie entre les deux jeux de données d'évaluation

Model	ROUGE Score (F) ($R1, R2, RL$)	Copy % $R1$
RÉFÉRENCE	52.31 / 34.00 / 49.70	79.36
SELFSUP	55.08 / 36.76 / 52.43	83.72
BACKSUM	55.29 / 36.39 / 52.89	89.24
BOTH	**58.02** / **38.64** / **55.56**	90.77

TABLE 5.2 – Scores sur les données `public_meetings`

Model	ROUGE Score (F) ($R1, R2, RL$)	Copy % $R1$
RÉFÉRENCE	33.83 / 15.86 / 31.05	74.25
SELFSUP	37.94 / 19.16 / 34.86	78.61
BACKSUM	39.17 / 20.06 / 36.19	86.96
BOTH	**40.23** / **21.02** / **37.26**	88.03

TABLE 5.3 – Scores sur les données `valid`

(`public_meetings` et `valid`). À l'inverse, les prédictions (quel que soit le modèle) ne diffèrent que de 5 copy%.

Cela semble suggérer que les modèles ont des difficultés à s'adapter afin de se comporter de manière plus « abstractive » lorsque cela est nécessaire. En d'autres termes, les modèles n'arrivent pas à prédire le taux de copie à adopter en fonction des textes d'entrées.

Augmentation du taux de copie associée à l'usage des données synthétiques Les modèles tirant profit des données synthétiques obtenues par résumé inverse, c'est-à-dire BACKSUM et BOTH présentent un taux de copie sensiblement supérieur aux autres modèles. En particulier, on obtient 6.44% de copie en plus au sein des résumés générés par BACKSUM par rapport à SELFSUP.

ROUGE et le taux de copie. Nos résultats suggèrent que les meilleurs modèles, en termes de score ROUGE sont ceux avec le plus haut taux de copie. Ceci pose la question de quel métrique nous souhaitons *vraiment* optimiser. D'une part, nous souhaitons maximiser ROUGE dans le but d'accroître la similarité avec le résumé de référence, d'autre part nous souhaitons générer des prédictions abstractives, et donc converger vers le taux de

copie de référence. En l'état, ces deux objectifs semblent en opposition, rendant difficile la tâche de déterminer quel modèle est le plus pertinent. Par exemple, dans les résultats sur l'ensemble de test (table 5.2) le modèle BackSum a des performances similaires à SelfSup en termes de score ROUGE mais cela s'accompagne d'un taux de copie bien supérieur, ce qui pourrait être pénalisé lors de notre jugement.

Conclusion Afin de contourner les contraintes liées à l'alignement des données, nous avons présenté deux approches permettant de tirer profit du grand nombre de comptes rendus non-alignés à notre disposition.

D'une part, nous avons pré-entraîné nos modèles de manière auto-supervisée à partir des comptes rendus non-alignés, avant de finir l'entraînement avec les données alignées. D'autre part, nous avons introduit le *résumé inverse* permettant de générer des transcriptions synthétique pour chaque compte rendu non aligné afin d'augmenter considérablement notre quantité de données.

Chacune des approches a montré des résultats encourageants, dépassant significativement le modèle de référence. De surcroît, la combinaison des deux approches nous a permis d'obtenir de meilleurs résultats encore, atteignant +6/+5/+6 ROUGE-1/2/L par rapport au modèle de référence.

Chapitre 6

Réduction du biais d'extractivité grâce aux tokens de contrôle

On trouve au sein des comptes rendus Ubiqus des prises de paroles de différentes natures : des discours préparés, longs et bien rédigés autant que des discussions plus courtes et hasardeuses. Ces prises de paroles hétérogènes présentent par conséquent une grande variance dans leurs longueurs et dans le degré de correction et de synthèse qu'elles nécessitent. Au contraire, les modèles de compte rendu automatique convergent vers un comportement trop extractif : les comptes rendus générés présentent un biais d'extractivité et une moindre variance.

Nous travaillons dans ce chapitre à réduire ce biais et à rendre les modèles plus robustes à cette variance en les entraînant à prédire explicitement la longueur et le taux de copie du résumé cible.

Table des matières

6.1 Problématique

À l'issue des travaux du chapitre 5 sur l'usage des données non-alignées, nous avons constaté la difficulté de comparer les modèles de résumé automatique à cause de contradictions dans nos objectifs. D'une part, nous souhaitons obtenir des résumés de qualité, ce que l'on mesure par la similarité lexicale avec les résumés cibles grâce à la métrique ROUGE. D'autre part, nous souhaitons générer des résumés *abstractifs*, et donc limiter la copie au sein des résumés. Or, nous observons dans les résultats (tableaux 5.2 et 5.3) une corrélation entre le score ROUGE et le taux de copie : les meilleurs modèles en termes de score ROUGE sont aussi ceux avec le taux de copie le plus élevé. Aussi, nous observons que les modèles ne s'adaptent pas, ou très peu, au taux de copie attendu. En effet, le taux de copie des prédictions effectuées à partir du jeu de validation et du jeu de test sont proches (autour de 5% de différence), alors qu'on observe un écart de 20% en absolu au sein des comptes rendus de rédacteurs humains qui constituent les données de référence (*cf.* tableau 5.1).

Nous formulons deux hypothèses :

(i) Nos modèles de résumé automatique sont biaisés en faveur d'un comportement *extractif* se traduisant par un taux de copie dans les prédictions plus élevé que dans les données de référence.

(ii) Nos modèles de résumé automatique ne parviennent pas (ou peu) à identifier le taux de copie à adopter et à adapter leur prédiction en fonction. En d'autres termes, le taux de copie observé dans les résumés automatiques présente une variance trop faible.

Nous pouvons observer ces deux hypothèses graphiquement grâce à la figure 6.1 où l'on présente la distribution des taux de copie par déciles des modèles de résumé automatique du chapitre 5 par rapport aux données de référence (`valid`). Les déciles sont calculés sur les données alignées automatiquement et manuellement (`man` et `auto` du chapitre 4). Les données de validation (`valid`) proviennent des données alignées (de même que `man` et `auto`), on observe logiquement que le taux de copie est également réparti : la densité est autour de 0.10 pour chaque décile.

Au contraire, les taux de copie observés au sein des résumés automatiques ne sont pas également répartis. En particulier on observe l'hypothèse (i) qui se traduit graphiquement par une distribution non centrée ; et l'hypothèse (ii) qui se traduit par un pic très important, au lieu d'une courbe presque plate comme pour `valid`.

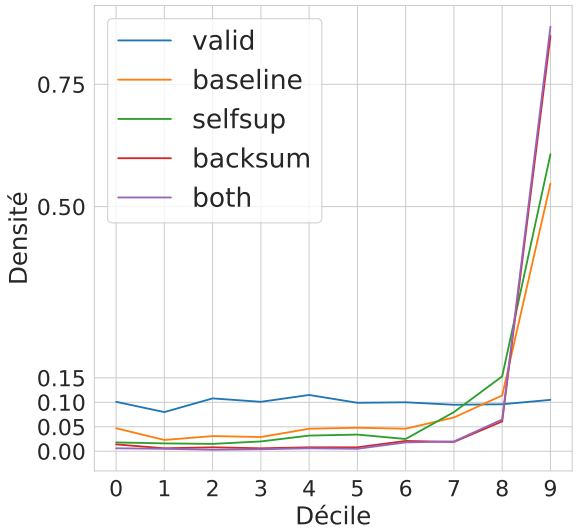


FIGURE 6.1 – Distribution du taux de copie par déciles (calculés sur les données d'entraînement) observé sur l'ensemble `valid` dans les résumés générés par les modèles de (TARDY *et al.*, 2020b, *cf.* chapitre 5 et tableau 5.3)

Dans ce chapitre, nous proposons d'entraîner les modèles à prédire explicitement la longueur et le taux de copie des résumés cible. On cherche à évaluer la capacité pour un modèle à prédire, à partir du texte source, si le résumé attendu est court ou long, et s'il présente un taux de copie élevé ou non. Ces caractéristiques sont représentées de manière discrète sous forme de tokens spéciaux, comme détaillé en section 6.2. Nous étudions d'abord la prédiction de ces caractéristiques, sans génération de résumé en section 6.3, puis l'apprentissage joint de la prédiction des caractéristiques et de la génération de résumé en section 6.4. Enfin, la section 6.5 présente des extensions aux méthodes proposées.

6.2 Représentation discrète de caractéristiques du résumé

L'objectif de ces travaux est d'étudier d'une part la capacité d'un système de résumé automatique à détecter les caractéristiques souhaitées de la prédiction – la longueur du résumé et le taux de copie à adopter – et d'autre part sa capacité à prendre en compte cette information afin de générer un meilleur résumé.

Nous suivons l'approche de [FAN *et al.*, 2018] qui propose d'utiliser des mots spéciaux – les *tokens* de contrôle – afin de prédire, et contrôler certaines caractéristiques du résumé.

On peut utiliser par exemple un token spécial `LEN` pour représenter la longueur de la cible avec 10 valeurs différentes : de `LEN_0` (court) à `LEN_9` (long). Le token est ensuite placé dans la source et informe le modèle de la longueur du résumé souhaité. Ici, le token ne représente pas directement la longueur de la cible mais le décile auquel appartient cette valeur. Ainsi, le token `LEN_0` signifie que le résumé attendu fait partie des 10% les moins longs (par rapport à l'ensemble données d'entraînement). La répartition des tokens par décile permet notamment que les tokens soient également répartis dans les données d'apprentissage, évitant ainsi un biais de représentation.

Plus formellement, les tokens décrivent des *caractéristiques* du résumé cible, c'est-à-dire des fonctions f_t qui associent un nombre réel aux éléments de nos jeux de données (les paires *(transcription, compte rendu)*). Pour un jeu de données $\boldsymbol{\mathcal{D}}$, et afin de représenter la caractéristique $f_t : \boldsymbol{\mathcal{D}} \mapsto \mathbb{R}$ par T tokens, on procède en plusieurs étapes :

1. On calcule, pour toute paire $(x_i, y_i) \in \boldsymbol{\mathcal{D}}$ la valeur de la caractéristique $v_i = f_t(x_i, y_i)$.
2. On détermine les $(T-1)$ quantiles q_j, $1 \leq j \leq T-1$ sur l'ensemble des données d'entraînement $\boldsymbol{\mathcal{D}}$. Par exemple, pour $T = 4$ on calcule les trois quartiles q_1, q_2 (la médiane) et q_3.
3. On définit les *valeurs de seuil* τ_j tels que $\tau_0 = -\infty$ et $\tau_j = q_j$ pour $j \in [\![1; T-1]\!]$
4. Enfin, on obtient la valeur du token associé à la paire i comme étant la plus grande valeur de j inférieure à la valeur de la caractéristique v_i ; c'est-à-dire : $token(x_i, y_i) = \arg\max_{j \in [\![1;T-1]\!]} \{v_i > \tau_j\}$.

Dans le cas général où un modèle génère un résumé et un token à partir d'un texte source x_i, on note $\hat{y}_i$ le résumé généré, et y_i le résumé de référence. De même, on note t_i le token de référence, c'est-à-dire $t_i = token(x_i, y_i)$ et $\hat{t}_i$ le token correspondant à la caractéristique t au sein du résumé généré : $\hat{t}_i = token(x_i, \hat{y}_i)$. Enfin, on note $\tilde{t}_i$ le token prédit par le modèle.

On cherche alors à maximiser trois objectifs :

— **Obtenir le bon token** (acc_t) : le token prédit correspond au token référence *i.e.* $acc_t = acc(\tilde{t}, t)$.

— **Obtenir un résumé avec la bonne caractéristique** ($\hat{acc}_t$) : la caractéristique au sein du résumé généré correspond à la caractéristique de référence *i.e.* $\hat{acc}_t = acc(\hat{t}, t)$.

— **Avoir un résultat cohérent** (coh_t) : la caractéristique du résumé généré correspond au token généré *i.e.* $coh_t = acc(\tilde{t}, \hat{t}_i)$.

On utilise pour la mesure de l'exactitude (*acc*) la fonction ci-dessous (eq. 6.1). De plus, pour chaque mesure d'exactitude (*acc*) nous pouvons également mesurer les distances (*dist*) telles que formuler dans (eq. 6.2).

$$acc(t, t') = 1/N \times |\{t_i \mid t_i = t'_i\}| \tag{6.1}$$

$$dist(t, t') = 1/N \times \sum_i |t_i - t'_i| \tag{6.2}$$

En pratique on considère les caractéristiques de *copie*, $f_c(x_i, y_i) = copy\%(x_i, y_i) = \text{ROUGE}_R(x_i, y_i)$, et de *longueur* $f_l(x_i, y_i) = |y_i|$.

6.3 Expériences préliminaires : prédiction du taux de copie

6.3.1 Présentation de la tâche

Faisant suite aux discussions de la section 5.5 concernant la difficulté des modèles à générer des résumés dont le taux de copie soit adéquat, nous proposons en expérience préliminaire d'étudier la prédiction du taux de copie du résumé cible, à partir du texte source.

L'idée est que, contrairement aux données issues d'articles de journaux par exemple (voir tableau 6.1 et section 1.3.2), les comptes rendus font preuve de grande variabilité quant au taux de copie que l'on y retrouve. Ceci s'explique par plusieurs aspects. D'abord, la nature des prises de paroles au sein d'une réunion peut varier. En particulier, la prise de parole peut adopter un style presque écrit, par exemple lorsque l'interlocuteur prononce un texte préparé ou un style oral lorsqu'il s'agit de questions-réponses. De ce fait, le discours sera plus ou moins bien structuré, plus ou moins bon grammaticalement et donc nécessitera plus ou moins d'intervention humaine pour en rendre compte. D'autre part, les transcriptions dont nous disposons sont des transcriptions automatiques. Celles-ci ne sont donc pas exemptes d'erreur. Suivant le contexte, le bruit, la qualité du micro et la distance de l'intervenant, la transcription sera donc de qualité variable, et nécessitera donc plus ou moins de correction.

Par conséquent, nous formulons l'hypothèse que la source (= la transcription), fournit des indices quant au taux de copie attendu dans le compte rendu. Un rédacteur humain pourrait donc par exemple prédire le niveau de fidélité à adopter par rapport à la trans-

cription en évaluant la qualité de la langue et de son organisation.

Il nous semble alors pertinent d'évaluer dans quelle mesure un modèle similaire aux modèles de résumé automatique parvient à prédire le taux de copie. Pour être tout à fait comparable expériences des chapitres précédents, nous utilisons strictement la même architecture, c'est-à-dire un modèle TRANSFORMER dont l'encodeur lit la source (la transcription) et le décodeur génère la prédiction. Dans cette tâche préliminaire, nous écartons la génération du résumé, et nous contentons de prédire le token relatif au taux de copie, donc une séquence d'un seul « mot ».

Les tokens sont représentés comme des « mots » normaux, la tâche s'apparente donc à de la classification en C classes, sans connaissances préalables de l'ordre de ces classes *i.e.* le modèle ignore a priori que $i < j \Rightarrow COPY_i < COPY_j$.

6.3.2 Cadre expérimental

Données Les ensembles de données UBIQUS (et CNN/DAILYMAIL pour comparaison) sont présentés avec leur taux de copie (moyen et les déciles) dans le tableau 6.1. Nous pouvons observer d'une part la différence importante du taux de copie moyen : 52.03% pour les données UBIQUS alignées contre 87.72% pour CNN/DAILYMAIL. De plus, il existe une plus grande diversité de taux de copie au sein des différentes paires dans les données UBIQUS : on a plus de 50% d'écart entre le premier décile (d_1) et le dernier (d_9) contre moins de 11% pour CNN/DAILYMAIL.

Nous menons l'expérience sur les données UBIQUS alignées manuellement, et automatiquement (`Man+Auto`).

Jeux de Données	**Déciles**					**Moyenne**
	d_1	d_3	d_5	d_7	d_9	
CNN/DailyMail	76.78	84.62	**89.08**	92.78	97.06	87.72
Ubiqus Man+Auto	23.48	41.10	**53.85**	65.00	77.14	52.03
Ubiqus Back	34.38	54.05	**64.00**	71.62	80.30	60.37

TABLE 6.1 – Déciles et moyenne des taux de copie au sein des différents jeux de données UBIQUS et CNN/DAILYMAIL

Métriques Afin d'évaluer la capacité des modèles à prédire le taux de copie souhaité à partir de la source nous mesurons l'exactitude (acc_t) et la distance absolue moyenne

($dist_t$) telles que définie en section 6.2 (en particulier dans les équations 6.1 et 6.2).

Modèles de Référence Les expériences des chapitres précédents n'utilisent pas de tokens, la comparaison directe n'est donc pas possible. Toutefois, nous pouvons mesurer si les taux de copie présents dans les comptes rendus automatiques correspondent à ceux des références (notés $\hat{c}_i$, *cf.* 6.2). On compare en quelques sortes une prédiction explicite du taux de copie via les tokens à une prédiction implicite via les caractéristiques des résumés.

Nous mesurons alors ces valeurs pour les modèles de résumé automatique issus de [Tardy *et al.*, 2020b] (chapitre 5) et les présentons dans le tableau 6.2.

Modèle	**Moyenne** $\bar{c}$	**Exactitude** ↑ $\hat{acc}_c$	**Distance** ↓ $\hat{dist}_c$	**Taux copie** *copy*%
Base	7.19	14.20%	3.304	74.25%
SelfSup	7.80	12.90%	3.559	78.61%
BackSum	8.50	12.00%	4.096	86.96%
Both	8.65	11.60%	4.162	88.03%

Table 6.2 – Évaluation de l'exactitude et de la distance des tokens de copie (avec $C = 10$) des modèles résumé automatique issus de [Tardy *et al.*, 2020b] (chapitre 5) sur l'ensemble `valid`. Les valeurs de références sont $\bar{c} = 4.53$ et $copy\% = 55.38$.

Modèles Nous calculons la performance de prédiction des tokens de copie selon différentes variantes du modèle :

1. **small** : architecture plus petite (*i.e.* encodeur et décodeur mono couche, 2 têtes d'attention, couches cachées de 128 dimensions, au lieu de respectivement 6 couches, 8 têtes, et 512 dimensions).
2. **vslowlr** : un *learning rate* 100× plus faible.
3. **bart30** : introduction de bruit similaire à Bart [Lewis *et al.*, 2020] *i.e.* permutation des phrases remplacement de portion de texte avec $p = 0.3$ (de manière similaire à 5.2).
4. **Finetune** : l'entraînement du modèle s'effectue à partir du modèle pré-entraîné pendant $50,000$ itérations. Il s'agit du même modèle pré-entrainé que SelfSup (*cf.* 5.2).
5. **double** : deux fois plus de tokens *i.e.* $C = 20$, au lieu de $C = 10$.

6.3.3 Résultats

Les résultats sont présentés dans le tableau 6.3. On observe que les modèles parviennent à correctement prédire le token de copie dans 22.80% des cas, avec $C = 10$ et dans 17.10% des cas avec $C = 20$. Cette exactitude est sensiblement supérieure aux exactitudes $\hat{acc}_c$ des modèles de résumé de précédemment étudiés : 14.20% pour modèle Base et 12.90% pour SelfSuf, *cf.* tableau 6.2).

Notons que l'espérance d'un modèle qui prédirait aléatoirement le token de copie serait, pour l'exactitude acc_c de respectivement $1/10 = 10\%$ et $1/20 = 5\%$ dans les cas $C = 10$ et $C = 20$. Par conséquent, par rapport à cette référence aléatoire, on remarque que l'exactitude avec $C = 10$ est 2.28 fois supérieure. l'exactitude avec $C = 20$ est quant à elle est 3.42 fois supérieure. Ceci pourrait indiquer que, en comparaison avec la difficulté de la prédiction, les modèles utilisant $C = 20$ tokens sont plus performants.

Cette expérience préliminaire nous confirme donc que les modèles utilisés ont bien la capacité d'apprendre à prédire le taux de copie de référence à partir de la source seule. Nous chercherons, dans les expériences suivantes, à tirer profit de la prédiction des tokens au sein de la génération des résumés, c'est-à-dire de produire des résumés qui comportent le bon taux de copie.

6.4 Prédiction, génération et contrôle du résumé

6.4.1 Problématique

Nous avons vu d'une part que les modèles de résumé automatique tendent à produire des textes avec des taux de copies élevés (voir tableau 6.2) et d'autre part qu'il est possible de prédire le taux de copie de référence à partir de la source avec plus d'exactitude que ce qui est fait implicitement lors de la génération des résumés.

Dans cette section, nous considérons des modèles entraînés à prédire les caractéristiques (la copie et la longueur) et à générer le résumé. Nous cherchons à étudier si la prédiction explicite de ces caractéristiques sous forme de tokens aide le modèle à générer des résumés qui présentent effectivement ces caractéristiques. Nous pourrons alors (i) comparer l'exactitude de la génération des tokens, par rapport aux expériences préliminaires ; (ii) comparer les caractéristiques des résumés par rapport aux autres modèles de résumé et enfin (iii) observer la cohérence entre les tokens prédits et les caractéristiques des résumés générés.

C	model	finetune	small	bart	vslowlr	it.	acc (%)
Simple ($C = 10$)	BASE	✗	✗	✗	✗	3.0	16.70
	+ *vslowlr*	–	–	–	✓	3.0	19.20
	BART	–	–	✓	✗	4.5	18.30
	+ *vslowlr*	–	–	–	✓	3.0	19.50
	SMALL	–	✓	✗	✗	3.5	18.40
	+ *vslowlr*	–	–	–	✓	5.0	17.90
	+ *bart*	–	–	✓	–	5.0	18.50
	FINETUNE	✓	✗	✗	✗	0.5	19.90
	+ *vslowlr*	–	–	–	✓	5.0	**22.80**
	+ *bart*	–	–	✓	–	4.5	22.00
Double ($C = 20$)	BASE	✗	✗	✗	✗	3.0	13.20
	+ *vslowlr*	–	–	–	✓	3.0	**17.10**
	BART	–	–	✓	✗	3.5	13.60
	+ *vslowlr*	–	–	–	✓	3.0	15.60
	SMALL	–	✓	✗	✗	1.0	14.20
	+ *vslowlr*	–	–	–	✓	5.0	12.40
	+ *bart*	–	–	✓	–	5.0	13.50
	FINETUNE	✓	✗	✗	✗	3.5	14.10
	+ *vslowlr*	–	–	–	✓	3.5	14.70
	+ *bart*	–	–	✓	–	4.0	14.70

TABLE 6.3 – Prédiction du taux de copie souhaité du compte rendu à partir de la source (transcription) en fonction des extensions du modèle de base (✓ : extension présente ; ✗ : extension absente ; – : même valeur que ligne du dessus).

Les tokens sont placés au début de la source, avant le texte de la transcription. Le modèle est ainsi *informé* des caractéristiques souhaitées. Durant l'entraînement, ces tokens sont masqués, c'est-à-dire remplacés par un token spécial `MASK` une fois sur deux. L'objectif est ainsi d'apprendre au modèle deux modes : le mode *informé* (`INF`) où le modèle dispose des tokens cibles, dans le but d'obtenir une prédiction avec ces caractéristiques ; et le mode *prédiction* (`PRED`) où l'on laisse le modèle prédire les caractéristiques souhaitées. Le mode prédiction est donc comparable avec les modèles des chapitres précédents car il n'exploite pas d'information supplémentaire. Le mode informé quant à lui nous permet de quantifier l'impact de ces informations sur la génération de résumés automatiques. De plus, ce mode offre à l'utilisateur un moyen de contrôler la génération.

6.4.2 Cadre Expérimental

Métriques et Références Les différents aspects étudiés par ces expériences sont mesurées par les métriques suivantes :

— Exactitude des tokens : acc_c.

— Exactitude des caractéristiques du résumé : $\hat{acc}_c$.

— Cohérence : coh_c.

— Taux de copie : *copy*%.

— Score du résumé : ROUGE.

Les métriques portants sur les résumés en eux-mêmes ($\hat{acc}_c$, *copy*%, ROUGE) sont utilisées pour l'évaluation par rapport aux modèles de résumé sans token *cf.* tableau 6.2). D'autre part, les valeurs des métriques portant sur les tokens (acc_c et coh_c) sont comparées aux résultats de l'expérience préliminaire *cf.* tableau 6.3.

Données Nous utilisons les mêmes données que pour les modèles de résumé de référence (issus du chapitre 5), à savoir (i) les données alignées (`man+auto`), les données non-alignées à des fins de pré-entraînements auto-supervisés (*cf.* section 2.5) et les données synthétiques `back` (*cf.* section 5.3).

Modèles Dans ces expériences, nous partons du modèle pré-entraîné SELFSUP introduit en section 5.2. Utiliser ce modèle comme base nous permet de réduire les temps de calculs grâce au pré-entraînement : seul le *fine-tuning* est effectué avec les tokens.

6.4.3 Résultats et Discussions

Résultats Le tableau 6.4 présente les résultats des modèles à différents stades de l'entraînement en nombre d'itérations de *fine-tuning*, à partir du modèle pré-entraîné durant 50,000 itérations. Les premières lignes concernent les modèles sans tokens : `NO-SELF` et `NO-BOTH` correspondants respectivement aux modèles SelfSup et Both du chapitre 5 (*cf.* 5.3). Viennent ensuite les résultats en mode *prédiction* (`PRED`) où le modèle n'a pas l'information du token de référence et doit donc le prédire ; puis le mode *informé* (`INF`) où le token est présent dans la source.

Mode	**It.**	**Tok. (%)** acc_c	**Abs. (%)** $\hat{acc}_c$	**Coh. (%)** $\hat{coh}_c$	**ROUGE(F)** $R1/R2/RL$	**Copie** $copy\%$
`NO-SELF`	4k	–	**12.9**	–	37.94 / 19.16 / 34.86	78.61
`NO-BOTH`	6k	–	8.30	–	**40.23 / 21.02 / 37.26**	88.03
`PRED`	6k	15.7	17.5	**38.0**	**34.60 / 15.18 / 31.46**	64.47
`PRED`	28k	17.0	**18.2**	37.4	34.25 / 14.66 / 31.10	62.30
`PRED`	4k	**18.4**	16.7	**38.0**	34.01 / 14.88 / 30.91	63.52
`PRED`	30k	18.0	17.5	36.2	33.55 / 14.08 / 30.57	**61.82**
`INF`	30k	93.9	**29.2**	**31.1**	37.20 / 15.55 / 33.76	**58.12**
`INF`	6k	96.1	24.7	25.5	**38.40 / 16.69 / 34.92**	59.96
`INF`	28k	92.7	26.5	29.0	37.55 / 15.85 / 34.04	59.03

Table 6.4 – Prédiction de tokens (mode `PRED`), génération de résumé et contrôle du taux de copie (mode `INF`) sur l'ensemble `valid` par rapport aux modèles de références sans tokens `NO-SELF` et `NO-BOTH`.

Meilleure maîtrise du taux de copie En mode prédiction (`PRED`), on observe un gain significatif en exactitude du taux de copie au sein des résumés ($\hat{acc}_c$) par rapport au modèle de référence `NO-SELF` : jusqu'à +5.30%. Ceci s'accompagne d'une baisse du taux de copie, passant de 78.61 à au plus 64.47%. À 30k itération, ce modèle atteint même 61.82% de copie, ce qui réduit nettement l'écart du taux de copie au sein des résumés automatique par rapport aux données (écart réduit de 23.23 à 6.44%) [1] .

1. Le taux de copie dans les données `valid` est de 55.38%. L'écart entre le taux de copie dans les résumés automatiques du modèle SelfSup (`NO-SELF`) est donc de $78.61 - 55.38 = 23.23\%$ contre $61.82 - 55.38 = 6.44;\%$ pour `PRED` après 30k itération.

Écart d'exactitude par rapport aux résultats de l'expérience préliminaire (sec. 6.3) Les résultats en prédiction (PRED) prédisent les tokens avec une exactitude proche – mais légèrement inférieure – aux résultats de l'expérience préliminaire : 18.2 contre 19.90% [2].

Score ROUGE inférieur en mode prédiction On observe une baisse du score ROUGE de plus de 3.34 R1 en mode prédiction de tokens (PRED). Ce résultat va dans le sens de l'hypothèse que nous avions formulée : le score ROUGE est corrélé avec le taux de copie.

Informer le modèle améliore sensiblement ses performances Enfin, on observe que les résultats en mode *informé* sont très nettement meilleurs sur chacune des métriques : le score ROUGE est plus de 3 point supérieur, l'exactitude de la copie au sein du résumé ($\hat{acc}_c$) est plus de 10% supérieure et enfin le taux de copie est inférieur d'environ 2%. Si ces résultats ne sont pas strictement comparables aux autres modèles puisqu'ils bénéficient d'informations supplémentaires, il demeure intéressant de noter qu'il s'agit du premier cas jusqu'alors où l'on observe une hausse du score ROUGE associée à une baisse du taux de copie.

De plus, les bonnes performances du mode informé pourraient encourager une approche en deux temps constituée d'un modèle de prédiction des tokens suivi de la génération du résumé en mode informé.

Augmenter la cohérence grâce à un critère explicite ? Que le taux de copie soit bien prédit – voire même que le modèle dispose de l'information (en mode informé) – ne signifie pas nécessairement que le résumé généré présente cette caractéristique : c'est ce que l'on mesure avec la cohérence. En particulier, il n'y a aucun élément explicite dans l'architecture ni dans la fonction de coût faisant le lien entre le token prédit et les caractéristiques du résumé, tout est implicite. Une piste intéressante pour de futurs travaux consisterait à étudier des méthodes pour renforcer explicitement la cohérence afin de rapprocher les tâches de prédiction et de génération. On pourrait par exemple penser à une fonction de coût basée sur la différence entre le token de copie prédit et le taux de copie constaté au sein du résumé généré.

2. On compare ici le meilleur modèle PRED avec les résultats du modèle FINETUNE de 6.3 qui est similaire.

6.5 Expériences Supplémentaires

Nous explorons dans cette section davantage de variantes de modèles avec tokens. Nous décrivons d'abord les extensions du cadre expérimental avec de nouveaux modèles et l'ajustement des métriques (6.5.1) avant de présenter et de discuter les résultats obtenus (6.5.2).

6.5.1 Cadre Expérimental

Modèles supplémentaires

— **Modèles *doubles* (prefix `D-`).** Nous introduisons les modèles *double* – noté avec le prefix `D` – utilisant $C = 20$ tokens pour le taux de copie. Ce choix est motivé par les résultats encourageants obtenus par cette variante lors des expériences préliminaires (6.3).

— **Prédiction uniquement (*OnlyPRED*, `OPRED`).** Dans la section précédente (6.4) nous avons étudié l'apprentissage conjoint de la prédiction, de la génération et du contrôle. Nous introduisons ici une variante sans contrôle durant l'apprentissage, c'est-à-dire uniquement en mode prédiction afin d'observer l'impact sur les performances isolément.

— **Apprentissage avec tokens et données synthétiques (suffixe `BOTH`).** Nous combinons le modèle BOTH du chapitre 5 (avec pré-entraînement auto-supervisé et *finetuning* sur les données synthétiques) et l'entraînement avec prédiction de tokens (`OPRED`). En d'autres terme, ce modèle est correspond à `D-OPRED` entraîné avec, en plus des données alignées, les données synthétiques de 5.3.

Ajustement des métriques Afin de comparer les modèles utilisant des nombres de tokens différents $C \in \{10; 20\}$ nous proposons un ajustement simple *adj_acc* des mesures d'exactitude :

$$\begin{aligned} adj(t^{20})_i &= \left\lfloor \frac{t_i^{20}}{2} \right\rfloor \\ adj\_acc(t^{20}, t^{10}) &= acc\left(adj(t^{20}), t^{10}\right) \\ &= 1/N \times \left|\left\{t_i^{20} \mid adj(t^{20})_i = t_i^{10}\right\}\right| \end{aligned} \tag{6.3}$$

Ceci revient à faire correspondre chaque token des modèles avec $C = 10$ (*i.e.* t^{10}) à deux tokens pour un autre modèle avec $C = 20$ (*i.e.* t^{20}). En effet, les tokens étant déterminés par des quantiles, le token 0 pour $C = 10$ – le premier décile – correspond aux tokens 0 et 1 pour $C = 20$, les deux premiers 20-percentiles.

Dans la suite, nous rapportons donc les deux valeurs : exactitudes (au niveau tokens, résumé et cohérence) ainsi que leurs valeurs ajustées.

6.5.2 Résultats et Discussions

Résultats Les résultats sont présentés dans le tableau 6.5. Ce tableau reprend les résultats précédents (tableau 6.4) en y ajoutant résultats des expériences de cette section.

Choix du nombre de tokens pour le taux de copie : 10 ou 20 ? On constate que le passage de 10 à 20 tokens s'accompagne d'une légère hausse du taux de copie ainsi que du score ROUGE, que ce soit en mode prédiction ou informé. En d'autres termes, les modèles avec 20 tokens proposent un nouveau compromis entre copie et ROUGE, qui se situe entre les modèles sans tokens et ceux avec 10 tokens. Il est intéressant de noter que le modèle `D-PRED` ne se trouve qu'à 1.94 points ROUGE de sa référence `NO-SELF` tout en restant 10% inférieur en termes de taux de copie. De plus, `D-INF` parvient à dépasser les deux références `NO-SELF` et `NO-BOTH` en score ROUGE tout en conservant un taux de copie respectivement 12% et 22% inférieur.

Entraînement en mode prédiction : `D-OPRED`. On observe une plus grande variance des résultats du modèle `D-OPRED` durant l'entraînement (d'une itération à l'autre). En particulier, celui-ci est plus extractif que `D-PRED` en début d'entraînement (72.33% de copie à 4k itération contre 68.97%) mais tend progressivement vers moins de copie (63.15% contre 67.76%).

Extensions du modèle Both avec 20 tokens. La combinaison du modèle BOTH – le meilleur modèle en score ROUGE du chapitre 5 – et de $C = 20$ tokens en mode prédiction obtient les meilleurs résultats. En effet, ce modèle dépasse sa référence `NO-BOTH` en termes de score ROUGE tout en réduisant le taux de copie de 10%.

Ce modèle est entraîné en mode prédiction uniquement : il est donc en tous points comparable à `OPRED` à l'exception des données d'entraînement. Il est intéressant d'observer que l'ajout de données synthétiques (`back` de 5.3) produit des résultats similaires que ce

Mode	**It.**	**Tok. (%)** acc_c	**Abs. (%)** $\hat{acc}_c$	**Coh. (%)** $\hat{coh}_c$	**ROUGE(F)** $R1/R2/RL$	**Copie** $copy\%$
NO-SELF	4k	–	**12.9**	–	37.94 / 19.16 / 34.86	78.61
NO-BOTH	6k	–	8.30	–	**40.23 / 21.02 / 37.26**	88.03
PRED	6k	15.7	17.5	**38.0**	**34.60 / 15.18 / 31.46**	64.47
PRED	28k	17.0	**18.2**	37.4	34.25 / 14.66 / 31.10	62.30
PRED	4k	**18.4**	16.7	**38.0**	34.01 / 14.88 / 30.91	63.52
PRED	30k	18.0	17.5	36.2	33.55 / 14.08 / 30.57	**61.82**
INF	30k	93.9	**29.2**	**31.1**	37.20 / 15.55 / 33.76	**58.12**
INF	6k	96.1	24.7	25.5	**38.40 / 16.69 / 34.92**	59.96
INF	28k	92.7	26.5	29.0	37.55 / 15.85 / 34.04	59.03
INF	4k	**97.4**	25.6	26.2	37.80 / 16.20 / 34.42	60.14
D-PRED	4k	10.8 **19.6**	9.2 **17.5**	22.2 **38.5**	**36.06 / 17.11 / 33.31**	68.97
D-PRED	8k	9.8 17.6	8.5 15.8	23.1 37.2	35.24 / 16.14 / 32.36	**67.76**
D-INF	4k	95.5 95.5	12.2 21.1	12.0 20.7	**40.75 / 19.18 / 37.39**	66.06
D-INF	6k	95.5 95.5	13.6 **25.0**	13.5 **24.7**	39.46 / 17.98 / 36.15	63.08
D-INF	8k	95.5 95.5	13.2 24.0	13.0 23.5	39.16 / 17.49 / 35.73	**61.41**
D-OPRED	4k	8.1 16.1	9.1 15.9	27.7 44.8	**37.82 / 18.15 / 34.84**	72.33
D-OPRED	22k	9.3 17.7	12.8 **20.5**	21.5 **34.4**	34.14 / 14.82 / 31.17	63.56
D-OPRED	20k	9.9 **18.4**	9.6 17.7	20.7 34.0	34.47 / 15.05 / 31.59	**63.15**
D-BOTH	14k	9.8 18.7	8.0 13.7	35.5 47.8	**40.43 / 20.45 / 37.48**	78.32
D-BOTH	18k	10.0 18.4	9.2 **15.6**	35.2 **48.7**	39.18 / 19.69 / 36.21	77.42
D-BOTH	26k	10.2 **19.0**	7.7 13.8	31.1 42.2	38.69 / 19.43 / 35.81	**76.45**

TABLE 6.5 – Prédiction, génération et contrôle sur l'ensemble `valid` des modèles avec $C = 10$ tokens (ou $C = 20$, préfix `D-*`) en mode prédiction (`PRED`) et informé (`INF`) par rapport aux modèles de références sans tokens `NO-SELF` et `NO-BOTH`. Pour les modèles `D-*`, la ligne du bas concerne les valeurs ajustées (*cf.* 6.5.1).

soit entre `D-OPRED` et `D-BOTH` ou entre `NO-SELF` et `NO-BOTH`. Toutefois, l'accroissement de la copie associé à l'usage des données synthétique (*cf.* 5.5) est amoindris en présence de tokens.

Ces résultats sont très encourageants car ils montrent que les données synthétiques permettent des gains de performances également sur les modèles avec tokens, et que ceux-ci parviennent à réduire l'accroissement du taux de copie lié à l'usage des données synthétiques.

6.6 Conclusion

Dans ce chapitre, nous avons présenté une approche simple pour la réduction du biais d'extractivité grâce à une meilleure prédiction du taux de copie souhaité au sein des résumés. Nous proposons d'utiliser des tokens de contrôle afin d'apprendre conjointement à contrôler les caractéristiques de la génération et à prédire les caractéristique, en l'absence de contrôle. Les modèles entraînés avec ces tokens génèrent des résumés plus proches des références en termes de taux de copie. De plus, les performances en termes de score ROUGE sont proches des modèles de référence, voire supérieures en mode informé (avec les informations de longueurs et de taux de copie) alors que ces deux objectifs (le taux de copie et ROUGE) se montraient contradictoires lors des travaux précédents. Ces résultats sont encourageants mais mettent en avant les difficultés que l'on rencontre à tirer des conclusions : nous obtenons des modèles présentant des caractéristiques bien distinctes en termes de score ROUGE ou de taux de copie, mais il est difficile de classer les modèles finement. En d'autres termes il est difficile de déterminer le meilleur équilibre entre ces deux métriques. En réponse à ce manque de visibilité quant aux performances de nos modèles, nous envisageons de soumettre les résumés automatiques à l'évaluation humaine au cours du chapitre 7.

Chapitre 7

Évaluation des comptes rendus automatiques et du cadre expérimental

Si les métriques automatiques permettent d'obtenir un aperçu des caractéristiques d'un compte rendu – par exemple de sa similarité avec la référence avec ROUGE, ou de son caractère abstractif avec le taux de copie – le degré d'adéquation entre ces métriques et le jugement humain demeure incertain.

Nous proposons alors dans ce chapitre une méthodologie permettant de recueillir le jugement humain dans le cadre de l'évaluation de comptes rendus automatiques. Cette évaluation nous permet de comparer les modèles entres eux, mais aussi d'évaluer la pertinence de notre cadre expérimental c'est-à-dire des métriques et données utilisées lors de nos évaluations automatiques.

Table des matières

7.1 Problématique

Les travaux des chapitres 4 et 5 ont consisté à développer des techniques permettant d'étendre l'apprentissage à un plus grand ensemble de données, que ce soit au travers de l'alignement automatique (chap. 4) ou de la valorisation des données non-alignées (chap. 5). Chacune de ces approches a été validée expérimentalement en évaluant la différence de performance des modèles de résumé automatique. Les gains de performances ainsi observés sont accompagnés d'une augmentation du taux de copie au sein des prédictions : c'est le *biais d'extractivité*. L'apprentissage et la prédiction de caractéristiques du résumé telles que la longueur et le taux de copie, tel que présenté dans le chapitre 6 a montré des résultats encourageants. Cependant, la réduction du biais d'extractivité s'accompagne d'une baisse du score ROUGE ce qui rend difficile l'interprétation des résultats. En particulier, on peut se demander s'il est souhaitable de réduire de biais d'extractivité si cela induit une baisse du score ROUGE, ou plus généralement, dans quelle mesure il est pertinent de sacrifier une métrique en faveur de l'autre.

C'est afin de trancher ces questions, et d'apporter un éclairage sur notre cadre d'évaluation que nous décidons de recourir à l'évaluation humaine. Dans ce chapitre, nous présentons l'évaluation humaine que nous avons mené : les problématiques rencontrées, les critères retenus et le cadre expérimental (sec. 7.2). Ensuite, nous discutons de l'interprétation des résultats dans le cadre de l'évaluation des différents modèles (7.5) et enfin l'interprétation des résultats par rapport à l'évaluation des métriques et du cadre expérimental du résumé automatique (7.6).

7.2 Évaluation humaine de comptes rendus automatiques

7.2.1 Difficultés de l'évaluation de comptes rendus

Évaluer la qualité d'un compte rendu n'est pas trivial. Au sein d'Ubiqus, la qualité des comptes rendus est contrôlée par des *relecteurs*. Ces derniers sont amenés quotidiennement à juger de l'acceptabilité des comptes rendus, et à les corriger si nécessaire. En revanche, leur travail n'est pas de les *comparer*, c'est-à-dire de déterminer le meilleur compte rendu parmi un ensemble. Or, dans notre cas, nous souhaiterions soumettre différents comptes rendus générés par nos modèles et les hiérarchiser par ordre de *qualité* (notion qu'il reste

à définir). De plus, le travail des relecteurs consiste à traiter des documents issus de rédacteurs professionnels, donc supposés déjà très bons.

Ainsi, le cadre d'évaluation des comptes rendus automatiques est bien différent du travail ordinaire des relecteurs. Nous ne souhaitons pas savoir si les comptes rendus sont acceptables ou non mais plutôt pouvoir ordonner les modèles de résumé automatique en fonction de la qualité des comptes rendus produits, y compris en dessous du seuil d'acceptabilité. Aussi, nous souhaitons distinguer différents critères de qualité du résumé afin d'en apprendre plus sur les différences qualitatives entre les modèles de résumé automatique.

Il est important de prendre en compte la subjectivité d'une telle évaluation, et d'essayer de la réduire autant que possible. On parle alors *d'accord inter-annotateur*, c'est-à-dire de parvenir à obtenir un consensus parmi les différents annotateurs, et *d'accord intra-annotateur* qui est le fait pour un annotateur de donner une même réponse à une même question posée à différents instants. Par exemple, sans barème ni critères particuliers, il pourrait être délicat pour des annotateurs d'évaluer des comptes rendus sur une échelle de 1 à 10. Les différents annotateurs auraient alors des perceptions différentes de la qualité du compte rendu d'une part, et des valeurs de l'échelle.

7.2.2 Évaluation des comptes rendus par comparaison avec *BWS*

Cette difficultée à évaluer par rapport à un système de notation (appelé *rating scale*, *RS*) est étudié par [Kiritchenko & Mohammad, 2017] dans le contexte d'évaluation du sentiment d'une phrase. Ils observent alors qu'il est difficile d'obtenir des résultats reproductibles lorsque l'on demande aux annotateurs de noter sur une échelle de 1 à 10 : l'accord intra et inter annotateur est faible. Ils montrent alors que l'évaluation est plus pertinente lorsqu'il s'agit de comparer des éléments, c'est-à-dire de déterminer le meilleur et le moins bon élément parmi k propositions, ce qu'ils appellent Best-Worst Scaling (BWS). Les résultats présentés suggèrent également que l'écart entre l'évaluation *RS* et BWS est d'autant plus important que l'évaluation est complexe ou que le nombre d'annotations est faible. Or, l'évaluation de comptes rendus nous semble plus complexe que celle de sentiments dans des phrases – ne serait-ce que du fait de la longueur des comptes rendus.

Afin de comparer la pertinence des différentes échelles, [Kiritchenko & Mohammad, 2017] utilisent la mesure Split Half Reliability (SHR). Il s'agit d'une métrique couramment utilisée notamment dans le cadre d'expériences de psychologie et dont le but est d'évaluer la *reproductibilité* d'une expérience. Les auteurs argumentent la plus grande pertinence

de SHR par rapport aux mesures habituelles d'accord intra-annotateur (comme le kappa de Fleiss). En effet, dans le cadre d'une évaluation avec BWS, les désaccords entre annotateurs ne sont pas forcément des signaux négatifs. Par exemple, parmi quatre comptes rendus, A, B, C et D, si A et B sont de qualité similaire et tous deux largement meilleurs que C et D, alors, on peut s'attendre à ce que la moitié des annotateurs choisissent A comme le meilleur, et l'autre moitié B [1]. Si d'apparence, au niveau individuel, il semble y avoir un désaccord, globalement ces évaluations nous informent que $A \approx B$. La SHR évalue la corrélation des scores entre deux ensembles d'évaluation tirées aléatoirement. En d'autre terme, on sépare les évaluations en deux moitiés aléatoirement. On mesure ensuite les scores de chaque modèle indépendamment par rapport aux deux sous-ensembles d'évaluation et calculons enfin la corrélation entre les deux ensembles de scores. Cette opération est répétée, typiquement 100 fois, afin de réduire l'aléa de la mesure.

Nous choisissons donc d'utiliser BWS avec $k = 4$ (tel que suggéré par les auteurs) afin de maximiser la reproductibilité de nos résultats – mesurée avec SHR – tout en gagnant en robustesse vis-à-vis de la complexité de l'évaluation et du faible nombre d'annotations. L'évaluation consiste alors, pour toute transcription x_i, à présenter aux annotateurs $k = 4$ prédictions issues des M modèles sélectionnés pour être évalués. L'annotateur choisit ensuite, pour chacun des C critères, le meilleur modèle, et le moins bon.

Le score d'un modèle est calculé par la différence des fréquences à laquelle le modèle est le meilleur, et respectivement le moins bon, *i.e.* $BWS(i) = \%best_i - \%worst_i$.

7.2.3 Critères d'évaluation

Afin de mieux comprendre les différences de qualité entre les modèles de génération de comptes rendus, nous conduisons l'évaluation par rapport à plusieurs critères :

— **Grammaire** : la grammaire est correcte, et plus généralement le compte rendu ne contient pas de fautes.

— **Information** : Le compte rendu contient des informations pertinentes et correctes.

— **Formulation** : Le compte rendu est rédigé à la manière d'un *compte rendu exhaustif* Ubiqus c'est-à-dire qu'il emploie un langage écrit (et non oral comme dans la transcription), qu'il reformule ce qui doit l'être *etc.*.

1. On peut à minima s'attendre à ce que ce soit le cas en moyenne, et donc que cela tende à être vrai pour un nombre important d'annotations.

Nous avons essayé de proposer des critères permettant aux évaluateurs de comparer les comptes rendus plus finement. En effet, certains comptes rendus peuvent être bien écrits, en termes de grammaire, mais inexacts en terme d'information, ou avec un style très oral issu de la transcription. Il est donc nécessaire de dissocier les différentes qualités et défauts de chaque compte rendu afin d'obtenir une meilleure compréhension des différences qualitatives entre les modèles de génération et de rendre plus claire l'interprétation de l'évaluation.

7.3 Cadre Expérimental

Données Les paires de segments *(transcription ; compte rendu)* évaluées sont issues du jeu de données `valid`. Afin de faciliter l'évaluation, nous avons manuellement sélectionné les segments afin de ne garder que les plus pertinents. En effet, du fait d'erreur d'alignement – qu'il soit humain ou automatique – et d'éventuelles erreurs de transcription automatique, il se peut que le compte rendu ne corresponde pas parfaitement à la transcription. Ce cas de figure peut rendre difficile, même pour un évaluateur humain, de déterminer quel devrait être le compte rendu d'après la transcription.

À partir des données filtrées, on forme des *tuples* (i, m_1, m_2, m_3, m_4) associant la transcription x_i au i-ième compte rendu issu des modèles $m_1, \ldots, m_4$ (avec $m_j \in [\![1; M]\!]$). Ces *tuples* seront ensuite soumis aux évaluateurs.

Modèles Nous conduisons l'évaluation sur les modèles issus des chapitres 5) et 6. Afin d'obtenir un score de référence, nous soumettons aussi les comptes rendus de référence (les cibles) à l'évaluation. L'évaluation comprend ainsi les comptes rendus issus de :

1. TGT : le compte rendu de référence (produit par un rédacteur UBIQUS).
2. BASE : le modèle de base issu de l'alignement automatique (`man` + `auto` dans chap. 4), repris en tant que modèle de référence par la suite (notamment en chap. 5, résultats tab. 5.3).
3. SELFSUP : modèle avec pré-entraînement auto-supervisé (sec. 5.2).
4. BACKSUM : modèle entraîné avec des données synthétiques grâce au *résumé inverse* (sec. 5.3).
5. BOTH : modèle qui combine SELFSUP et BACKSUM.
6. 10TOKS(PRED) : modèle avec 10 tokens de copie et de longueur, en mode prédiction.

7. 10TOKS(INF) : modèle avec 10 tokens de copie et de longueur, en mode informé.
8. 20TOKS(PRED) : modèle avec 20 tokens de copie et 10 de longueur, en mode prédiction.
9. 20TOKS(INF) : modèle avec 20 tokens de copie et 10 de longueur, en mode prédiction.

Annotateurs Deux équipes d'annotateurs ont été formées au sein d'UBIQUS. Une première composée de 5 relecteurs, donc des professionnels de la validation de comptes rendus. La seconde comprend 4 membres de l'équipe UBIQUS LABS.

Les données présentées, c'est-à-dire une transcription d'une part et quatre comptes rendus d'autre part, sont les mêmes pour chaque membre d'une même équipe. Les deux équipes, quant à elles, traitent des données différentes.

Plateforme d'évaluation Une *évaluation* consiste à déterminer pour chacun des trois *critères* (*cf.* 7.2.3) lequel des quatre comptes rendus proposés est respectivement le meilleur, et le moins bon. Une plateforme d'évaluation est mise à disposition des évaluateurs. Son fonctionnement est présenté au travers d'un exemple en figure 7.1.

Chaque annotateur est associé à une *équipe*, et a un *nom d'utilisateur* et un *mot de passe* qui lui est propre. Une fois l'évaluation réalisée, il envoie les informations au serveur qui centralise les évaluations.

7.4 Analyse de l'évaluation

Les évaluations ont été menées par deux équipes de respectivement 5 et 4 personnes. Chacune des équipes a annoté 10 transcriptions. Dans les deux cas, les comptes rendus évalués sont tirés à partir de 9 sources : 8 modèles de résumé automatique ainsi que la cible (les comptes rendus de références).

Chaque *tuple* à évaluer présentant 4 comptes rendus différents, nous présentons 3 *tuples* pour chaque transcription afin d'évaluer chacune des 9 sources.Ainsi, pour chaque transcription i, on tire aléatoirement les modèles composant le premier tuple t_1 – *e.g.* $t_1 = (i, 4, 2, 5, 1)$ – ; puis un second t_2 parmi les modèles restant (privé de ceux de t_1) – *e.g.* $t_2 = (i, 9, 3, 7, 6)$. Le troisième *tuple* contient alors le seul modèle présent ni dans t_1 ni dans t_2, (*e.g.* 8), ainsi que 3 comptes rendus déjà présents, tirés aléatoirement, *e.g.* $t_3 = (i, 8, 3, 4, 5)$. Les comptes rendus de référence (*i.e.* TGT) sont exclus lors du tirage de t_3 afin de limiter la présence des comptes rendus de références à une par transcription pour donner

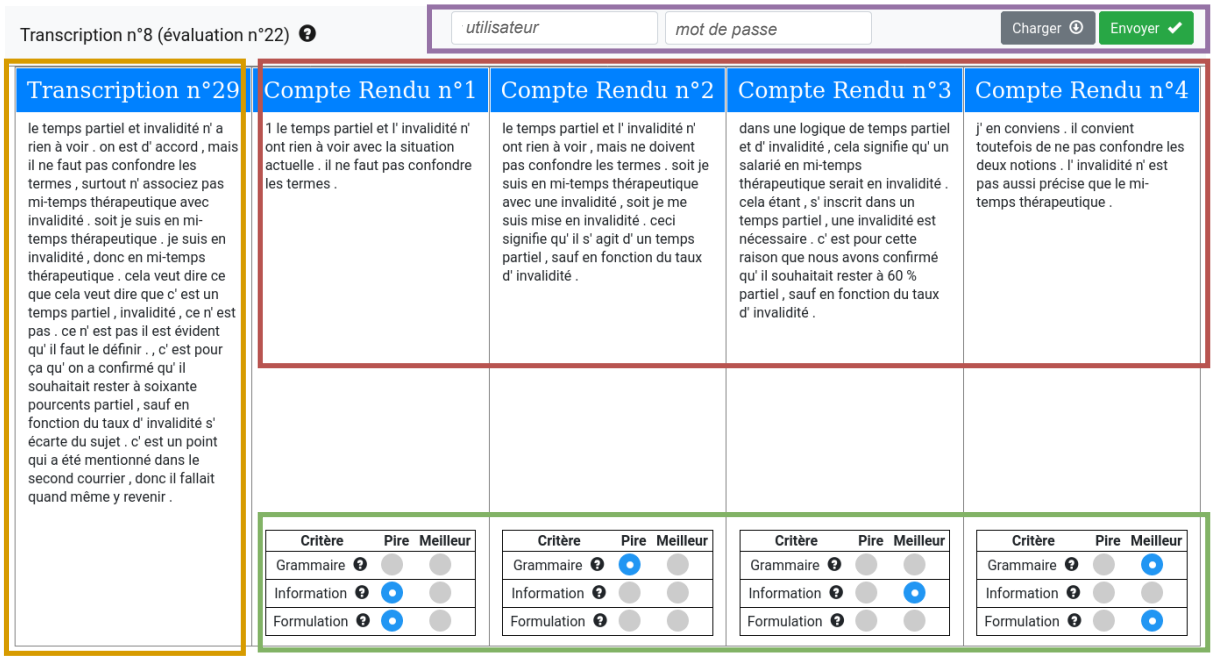


FIGURE 7.1 – Exemple d'évaluation sur la plateforme. Un annotateur saisit *son nom d'utilisateur* et son *mot de passe* afin de pouvoir charger une *évaluation* (encadré violet). Celle-ci consiste en une *transcription automatique* (encadré orange) et quatre *comptes rendus* (encadré rouge) tirés aléatoirement parmi les modèles (*cf.* sec. 7.3 §Modèles). L'évaluateur choisit alors, pour chacun des trois critères (*cf.* 7.2.3) le meilleur et le moins bon compte rendu (encadré vert)

davantage de place à l'évaluation des comptes rendus automatiques. Par conséquent, il y a exactement autant d'évaluation des comptes rendus que de transcriptions évaluées, c'est-à-dire 20 (10 par équipe). Le tableau 7.1 présente le nombre de *tuple* d'évaluation par paire de modèles.

	TGT	BASE	SELFSUP	BACKSUM	BOTH	PRED10	PRED20	INF10	INF20
TGT	20								
BASE	8	28							
SELFSUP	6	13	28						
BACKSUM	9	7	10	27					
BOTH	6	10	13	13	29				
PRED10	10	11	11	10	12	30			
PRED20	4	13	9	9	9	14	26		
INF10	8	11	7	13	10	11	12	25	
INF20	9	11	15	10	14	11	8	3	27

TABLE 7.1 – Nombre de *tuples* d'évaluation par paire de modèle. La valeur à la ligne i, colonne j représente le nombre de *tuples* contenant les modèles i et j. Les valeurs redondantes (symétriques par rapport à la diagonale, *i.e.* $(i, j) = (j, i)$) ne sont pas répétées.

Les 3 évaluations concernant une même transcription sont présentées aux annotateurs successivement afin de factoriser le temps nécessaire à l'évaluateur pour prendre connaissance de la transcription.

	Équipe 1			**Équipe 2**			**Global**	
	κ	r	ρ	κ	r	ρ	r	ρ
Grammaire	0.29	0.90	0.77	0.23	0.81	0.65	0.93	0.90
Information	0.38	0.89	0.83	0.19	0.77	0.66	0.91	0.91
Formulation	0.36	0.91	0.89	0.14	0.75	0.68	0.93	0.89
Moyenne	0.34	0.93	0.85	0.19	0.84	0.68	0.95	0.93

TABLE 7.2 – Accords inter-annotateurs (Fleiss κ) et *reproductibilité* (*SHR* : corrélations Pearson r et Spearman ρ) par équipes et par critères.

Afin de mesurer la pertinence des évaluations obtenues nous évaluons d'une part l'accord *intra-annotateur* avec le kappa de Fleiss (κ) et d'autre part la *reproductibilité* des évaluations via la SHR. Ces valeurs sont présentées dans le tableau 7.2. Le kappa de Fleiss est une mesure statistique de concordance lors de l'assignation qualitative d'*objets*

au seins de *catégories* par plusieurs *observateurs*. Dans notre cas, pour un *tuple* donné, les *catégories* correspondent aux quatre modèles présentés, et *objets* correspondent, aux valeurs « Meilleur » et « Pire » de chaque critère. On évalue alors, si les observateurs, ont, pour chaque critère sélectionné le même modèle pour chaque *issue* (*i.e.* « Meilleur » et « Pire »). Il y a donc, pour chaque *tuple*, $C \times 2 = 3 \times 2 = 6$ observations (nombre de critères multiplié par le nombre d'issues). De fait, cette valeur ne se calcule que pour des observateurs ayant des observations communes. Ceci explique l'absence de mesure κ sur l'ensemble des deux équipes (« Global » dans le tableau 7.2).

D'autre part la SHR mesure la *reproductibilité* de l'expérience. Cela consiste à séparer les évaluations en deux de manière égale et aléatoire. On considère ensuite les corrélations (r de Pearson et ρ de Spearman) entre les résultats obtenus sur les deux sous-ensembles d'évaluations. En d'autres termes, on mesure la corrélation entre les scores obtenus sur une moitié des évaluations et ceux obtenus sur l'autre moitié. Ce procédé ayant une part d'aléatoire, on utilise en pratique la moyenne des corrélations sur 100 tirages.

Comme discuté en section 7.2.2 (et sur le site[2] d'un auteur de Kiritchenko et Mohammad, 2017), on observe un faible accord inter-annotateur (entre 0.14 et 0.38) alors même que les corrélations de la SHR sont élevées ($[0.75, 0.93]$). Ceci suggère bien que le *désaccord* apparent des évaluateurs n'est pas une nuisance à la cohérence globale de l'évaluation.

Les différences en termes de reproductibilité et d'accord inter-annotateur au sein des deux équipes n'est pas négligeable, avec 0.15, 0.09 et 0.25 de différences concernant respectivement κ, r et ρ. Il est intéressant de noter cependant que, bien que l'équipe 2 présente a priori de moins bons résultats, les résultats globaux (sur les deux équipes) demeurent meilleurs que chacune des équipes prise indépendamment.

7.5 Évaluation des modèles

Comme suggéré dans [Kiritchenko & Mohammad, 2017] nous établissons un score pour chaque modèle i par *counting procedure* : $BWS\%(i) = \%Meilleur(i) - \%Pire(i)$. Les scores, ainsi que les proportions de « Meilleur » et de « Pire » sont reportés dans les tableaux 7.3, 7.4 et 7.5.

On observe tout d'abord une très large domination des comptes rendus de référence (TGT). Ce résultat est peu surprenant dans la mesure où il s'agit de comptes rendus

2. https://saifmohammad.com/WebPages/bwsVrs.html#SHR

	TGT	BASE	SELFSUP	BACKSUM	BOTH	PRED10	PRED20	INF10	INF20
Grammaire	**0.68**	-0.34	0.06	0.13	**0.21**	-0.14	-0.05	-0.36	-0.03
Information	**0.57**	-0.29	0.10	0.07	**0.31**	-0.20	-0.02	-0.32	-0.09
Formulation	**0.61**	-0.33	0.01	0.11	**0.27**	-0.18	0.00	-0.39	0.03
Moyenne	**0.62**	-0.32	0.06	0.10	**0.26**	-0.17	-0.02	-0.36	-0.03

TABLE 7.3 – Scores des modèles (valeurs dans $[-1; 1]$).

	TGT	BASE	SELFSUP	BACKSUM	BOTH	PRED10	PRED20	INF10	INF20
Grammaire	**0.72**	0.10	0.22	0.25	**0.41**	0.11	0.23	0.13	0.20
Information	**0.66**	0.10	0.25	0.32	**0.43**	0.10	0.26	0.10	0.14
Formulation	**0.70**	0.11	0.19	0.27	**0.41**	0.10	0.26	0.10	0.23
Moyenne	**0.69**	0.10	0.22	0.28	**0.42**	0.10	0.25	0.11	0.19

TABLE 7.4 – Proportion de « Meilleur » obtenus par modèles (valeurs dans $[0; 1]$).

	TGT	BASE	SELFSUP	BACKSUM	BOTH	PRED10	PRED20	INF10	INF20
Grammaire	**0.04**	0.45	0.16	**0.12**	0.20	0.25	0.28	0.49	0.23
Information	**0.09**	0.39	0.14	0.25	**0.12**	0.30	0.27	0.42	0.23
Formulation	**0.09**	0.44	0.18	0.17	**0.14**	0.27	0.26	0.49	0.20
Moyenne	0.07	0.42	0.16	0.18	0.15	0.28	0.27	0.47	0.22

TABLE 7.5 – Proportion de « Pire » obtenus par modèles (valeurs dans $[0; 1]$).

rédigés par des professionnels. Toutefois, ce score nous permet de mieux percevoir l'échelle des scores mesurés. En effet, si le score maximum théorique est de 1, il est intéressant de noter que même un compte rendu professionnel ne l'atteint pas. En particulier, les comptes rendus de références sont considérés Meilleurs dans 69% des évaluations contenant la référence, ce qui suppose qu'un compte rendu automatique est jugé meilleur que la références dans 31% des cas. D'autre part, TGT est évalué Pire dans 7% des évaluations. Ce résultat est plus étonnant, et est sûrement à imputer à des erreurs de segmentations et/ou d'alignement dans les données créant une inadéquation entre la transcription et le compte rendu.

Les résultats concernant les modèles développés au chapitre 5 viennent appuyer les conclusions tirées dans [Tardy *et al.*, 2020b] : on observe en effet un net gain des deux approches SelfSup et BackSum par rapport au modèle de référence BASE sur chacun des critères. De même, leur combinaison au sein du modèle Both obtient un score encore meilleur, confirmant la bonne synergie des deux approches.

On note que les résultats pour chaque critère sont proches de la moyenne. De plus, il ne semble pas se dégager de tendance quant au critère ayant le meilleur (ou le moins bon score). En particulier, nous n'observons pas de modèle s'étant « spécialisé » sur un critère c'est-à-dire présentant de grands écarts entre ses scores par critères. Ceci semble indiquer que, sur les modèles évalués, les différences de qualités se traduisent globalement, pas spécifiquement sur l'un des critères.

En revanche, les résultats sont moins encourageants concernant les modèles avec tokens. Que ce soit avec 10 ou 20 tokens, en mode prédiction (Pred) ou informé (Inf) aucun modèle ne surpasse leur modèle de référence : SelfSup. On note toutefois que le modèle Pred20 obtient autant ou plus de « Meilleur » que SelfSup ce qui peut s'avérer intéressant. En effet, dans le score calculé par *counting procedure*, les « Meilleur » et « Pire » s'annulent. Dès lors, un modèle étant « Meilleur » puis « Pire » obtient un score de 0 de même qu'un modèle qui ne serait ni « Meilleur » ni « Pire ». Suivant les contextes, on pourrait pourtant préférer un modèle plus conservateur (donc rarement « Pire », quitte à ne pas être « Meilleur » non plus) ou prenant davantage de risques (parfois « Meilleur » quitte à être aussi « Pire » parfois). En particulier, on remarque que, bien que le score de Pred20 soit moins bon que SelfSup, celui-ci demeure plus fréquemment « Meilleur » sur chaque critère. Cet aspect pourrait notamment être intéressant dans des contextes où les résumés automatiques sont filtrés. C'est par exemple le cas si l'on utilise le résumé automatique comme outil d'aide à la rédaction. Les résumés de qualité sont utilisés comme

point de départ et sont corrigés par un rédacteur. Ceci nécessite d'avoir un résumé de qualité suffisante pour que la correction soit possible et moins coûteuse que la rédaction. On ne considérerait alors que les résumés surpassant un certain seuil de qualité. En d'autres termes on se préoccuperait ici davantage du taux de « Meilleur » que du score global, ce qui change l'ordre des modèles évalués : on préférerait PRED20 à SELFSUP.

On constate de moins bons scores , que ce soit pour 10 ou 20 tokens, en mode informé par rapport au mode prédiction. Ce résultat est particulièrement étonnant dans la mesure où il va à l'encontre des résultats du chapitre 6 (qui donnait un net avantage aux modèles informés, en termes de score ROUGE) et surtout du fait que de tels modèles bénéficient d'informations supplémentaires lors de la génération. On peut supposer qu'il est difficile pour le modèle de réduire son biais d'extractivité et que ce faisant, il devient plus prompt aux erreurs.

Ce résultat peut aussi invalider l'option d'utiliser un modèle de prédiction de token en amont du modèle de résumé. En effet, si l'on constate de moins bons résultats en mode informé (donc avec les bons tokens à 100%), on peut douter qu'une meilleure prédiction de token apporte de meilleurs résultats.

7.6 Évaluation du cadre expérimental

Durant cette thèse, en particulier dans les chapitres 4, 5 et 6 nous avons évalué les performances des modèles de résumé automatique en termes de score ROUGE et de taux de copie sur des ensemble de test (*i.e.* `public_meetings`) et de validation (`valid`). En plus de mesurer la performance des modèles, l'évaluation humaine est l'occasion de tester la pertinence de ce cadre expérimental.

Évaluation des métriques On calcule les corrélations entres les métriques automatiques ROUGE, le taux de copie et les résultats de l'évaluation humaine (le score BWS). En particulier, on s'intéresse d'une part aux corrélations au sein des modèles sans tokens issus du chapitre 5 ; d'autre part au sein des modèles avec tokens, issus du chapitre 6 et enfin sur l'ensemble des modèles (*cf.* tableau 7.6). Ces mesures nous confirment les différences importantes entre les résumés produits par les modèles avec et sans tokens. Le score ROUGE est corrélé au jugement humain ainsi qu'au taux de copie pour les modèles *sans* tokens. Or, on constate l'inverse pour les modèles *avec* tokens : ROUGE se trouve décorrélé de BWS et du taux de copie. Il s'agit d'un signal important quant à

la difficulté à réduire l'extractivité des résumés automatiques : quand bien même nous parvenons grâce au tokens à diminuer le taux de copie sans trop perdre en performance en termes de score ROUGE, ce comportement n'est pas récompensé par les évaluateurs humains. D'autre part, la corrélation entre le score ROUGE et le jugement humain sur les modèles avec tokens, ainsi que sur l'ensemble des modèles (Global) est trop faible pour que l'on puisse considérer cette métrique pertinente dans le cadre de telles expériences. Ceci semble indiquer que, faute d'autres métriques plus fiables, l'évaluation humaine est nécessaire pour toute conclusion sur la performance des modèles. En particulier, le fait de travailler sur la réduction du biais d'extractivité semble avoir introduit au sein des modèles des comportements plus lourdement pénalisé par les évaluateurs humains que par le score ROUGE.

Modèles	**BWS & R1**		**BWS & Copy%**		**R1 & Copy %**	
	r	ρ	r	ρ	r	ρ
Sans Token (chap. 5)	0.88	0.80	0.64	1.00	0.87	1.00
Avec Tokens (chap. 6)	0.07	0.00	0.95	1.00	-0.17	0.00
Global	0.44	0.29	0.69	0.83	0.13	0.10

TABLE 7.6 – Corrélations Pearson (r) et Spearman (ρ) entre l'évaluation humaine (BWS), le score ROUGE-1 (R1) et le taux de copie (copy%) mesurées sur les modèles *sans* tokens (chap. 5) ; *avec* tokens (chap. 6) et l'union des deux (global)

Suggestions de métriques Le score ROUGE étant insuffisant pour mesurer la qualité des résumés automatiques nous proposons d'explorer d'autres métriques automatiques. En particulier, nous souhaitons évaluer la corrélation avec le jugement humain de métriques sans références, c'est-à-dire des métriques qui mesurent la qualité intrinsèque du compte rendu plutôt que sa similarité avec le compte rendu de référence. Nous proposons alors la *perplexité* et la *dual cross-entropy*. On mesure la perplexité à l'aide d'un modèle de langage (non neuronal) entraînés avec KENLM [HEAFIELD, 2011] sur l'ensemble des comptes rendus cibles des jeux de données `man`, `auto` et `back`. La perplexité $\text{ppl}_P(x)$ de la séquence $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ par rapport au modèle de langage P correspond à l'exponentiel de l'entropie $\text{H}_P(\boldsymbol{x})$ (*cf.* eq. 7.1, 7.2). En pratique, nous mesurons l'entropie, et la perplexité, par mot afin de ne pas pénaliser les longues séquences.

$$\mathrm{H}_P(\boldsymbol{x}) = -\frac{1}{|x|}\sum_{x_i}\log P(x_i \mid x_{<i}) \tag{7.1}$$

$$\mathrm{ppl}_P(\boldsymbol{x}) = 2^{\mathrm{H}_P(\boldsymbol{x})} \tag{7.2}$$

Pour un modèle de langage P donné, la perplexité sera d'autant plus faible que la séquences $\boldsymbol{x}$ est jugée vraisemblable par P, on cherche donc à minimiser la perplexité.

La deuxième métrique considérées est la *dual cross-entropy* de [Junczys-Dowmunt, 2018] qui mesure l'entropie croisée (*cross entropy*) de paires de séquences, par exemple $(\boldsymbol{x}, \boldsymbol{y})$ à partir de deux modèles de directions inverses *i.e.* A qui prédit $\boldsymbol{x}$ à partir de $\boldsymbol{y}$ et B qui prédit $\boldsymbol{y}$ à partir de $\boldsymbol{x}$. Cette mesure est notamment utilisée en traduction automatique où il est courant d'avoir des modèles dans les deux sens : par exemple français vers anglais ainsi que son inverse français vers anglais. Dans notre cas, il s'agit de paires de comptes rendus et de transcriptions, et, depuis la section 5.3 nous utilisons des modèles de résumés inverses : nous disposons donc bien de modèles pour les deux directions.

On cherche à minimiser la différence (en valeur absolue) de l'entropie croisée du résumé $\mathrm{H}_S(y \mid x)$ (*cf.* eq. 7.3) et celle du résumé inverse $\mathrm{H}_B(x \mid y)$ (*cf.* eq. 7.4), c'est-à-dire $\min|\mathrm{H}_S(y \mid x) - \mathrm{H}_B(x \mid y)|$. On souhaite également minimiser les entropies croisées dans chaque sens (de même que l'on cherchait à minimiser l'entropie dans le cadre de la perplexité), donc en particulier leur moyenne : $\frac{1}{2}(\mathrm{H}_S(y \mid x) + \mathrm{H}_B(x \mid y))$. À partir de ces deux membres, on obtient le score dual_xent de [Junczys-Dowmunt, 2018], eq. 7.6. Ce score prend ses valeurs dans $[0; 1]$ où 1 est le meilleur score, on cherche donc à maximiser la *dual cross-entropy*.

$$\mathrm{H}_S(y \mid x) = -\frac{1}{|y|}\sum_{y_i}\log P(y_i \mid y_{<i}; x) \tag{7.3}$$

$$\mathrm{H}_B(x \mid y) = -\frac{1}{|x|}\sum_{x_j}\log P(x_j \mid x_{<j}; y) \tag{7.4}$$

$$\mathrm{dual\_xent}(x, y) = \exp\left(-|\mathrm{H}_S(y \mid x) - \mathrm{H}_B(x \mid y)|\right. \tag{7.5}$$

$$\left. + \frac{1}{2}(\mathrm{H}_S(y \mid x) + \mathrm{H}_B(x \mid y))\right) \tag{7.6}$$

De même que pour les scores ROUGE, nous mesurons les corrélations de ces deux métriques avec le jugement humain (tableau 7.7). Les deux métriques – ppl et dual_xent

– obtiennent une meilleure corrélation avec le jugement humain par rapport à ROUGE [3]. On observe également que les corrélations entre jugement humain et perplexité sont très proches que l'on considère les modèles *avec* ou *sans* tokens. D'autre part, la *dual cross-entropy* présente la plus haute corrélation globale avec le jugement humain parmi les métriques considérées.

Modèles	**BWS & ppl**		**BWS & dual_xent**	
	r	ρ	r	ρ
Sans Token (chap. 5)	-0.91	-0.80	0.87	0.80
Avec Tokens (chap. 6)	-0.84	-0.80	0.48	-0.20
Global	-0.52	-0.50	0.76	0.79

TABLE 7.7 – Corrélations Pearson (r) et Spearman (ρ) entre l'évaluation humaine (BWS), la perplexité (ppl) et la *dual cross-entropy* (dual_xent) mesurées sur les modèles *sans* tokens (chap. 5) ; *avec* tokens (chap. 6) et l'union des deux (global)

Filtrage des données d'évaluation Durant la préparation de l'évaluation humaine des comptes rendus automatiques nous avons sélectionné manuellement les paires de segments *(transcription ; compte rendu)* avant des les soumette à l'évaluation. En effet, les données peuvent comporter des erreurs de transcription automatique ou d'alignement. Nous avons donc préféré ne soumettre que des paires de bonne qualité afin de ne pas rendre la tâche d'évaluation plus difficile pour les annotateurs.

Le même raisonnement peut s'appliquer aux évaluations automatiques, c'est-à-dire que l'on peut filtrer les jeux de données d'évaluation. En particulier, nous filtrons `valid` par pourcentage de meilleures paires en termes de *dual cross-entropy*. Le tableau 7.8 rapporte les valeurs de ROUGE et du taux de copie par modèles et par sous-ensemble en fonction du pourcentage de paires utilisées : 100% correspond donc au jeu d'évaluation `valid` dans son intégralité. De même, nous mesurons les corrélations entre le jugement humain (le score BWS), ROUGE (R1) et le taux de copie (copy%) dans le tableau 7.9.

On observe une hausse régulière du taux de copie ainsi que du score ROUGE pour chaque modèle à mesure que l'on filtre les paires, avec un écart en taux de copie de +15.44% dans les données et jusqu'à +17.76 R1 (pour SELFSUP sur `valid` filtré à 10%. Cette tendance est compréhensible dans la mesure où l'on attend du filtre qu'il sélectionne

3. La corrélation avec la perplexité est négative dans la mesure où c'est une métrique que l'on cherche à minimiser à l'inverse de BWS.

les paires comportant le moins d'erreurs de transcription et d'alignement et que ces erreurs tendent à rendre moins similaires les transcriptions par rapport aux comptes rendus. En effet, une transcription utilisant de mauvais mots (erreur de transcription) ou ne traitant pas du même sujet que son compte rendu (erreur d'alignement) aura un taux de copie plus faible, et le résumé obtenu sera moins similaire à celui attendu.

On observe par ailleurs que le modèle PRED20 – qui est déjà le plus proche du taux de copie sur les données non filtrées (*i.e.* à 100%) avec un écart de 4.58% – est celui qui suit du plus près l'évolution du taux de copie en finissant à +14.03% sur les meilleurs 10% contre +15.44% dans les données. Ce résultat met en avant la meilleure capacité de ce modèle à adapter son taux de copie aux données d'entrée, ce qui était une motivation de ce chapitre.

En revanche, étonnamment, les résultats en mode informé `INF` ne présentent pas un grand écart par rapport au mode prédiction `PRED`, surtout avec $C = 20$ tokens où l'on obtient un écart de 0.06 R1 entre PRED20 et INF20.

	TGT	BASE	SELFSUP	BACK	BOTH	PRED10	PRED20	INF10	INF20
100%	–	33.83	37.94	39.17	37.94	34.60	38.40	36.06	40.75
	55.38	74.25	78.61	86.96	79.99	64.47	59.96	68.97	66.06
90%	–	+1.14	+1.40	+1.48	+1.37	+1.21	+1.35	+1.39	+1.42
	+1.61	+1.46	+1.29	+1.15	+1.39	+1.46	+1.50	+1.66	+1.57
75%	–	+2.95	+3.28	+3.09	+3.32	+3.10	+3.12	+3.13	+3.09
	+3.84	+2.47	+2.30	+1.91	+2.63	+3.03	+3.79	+3.36	+3.53
50%	–	+6.03	+6.88	+6.29	+6.71	+6.06	+5.92	+5.87	+5.90
	+7.98	+3.92	+3.18	+2.52	+3.77	+4.95	+7.44	+4.68	+5.79
25%	–	+9.05	+11.53	+9.79	+11.70	+8.73	+9.30	+9.87	+9.16
	+12.14	+4.51	+4.64	+3.72	+5.82	+5.57	+10.53	+6.98	+7.55
10%	–	+10.78	+17.76	+14.46	+16.18	+10.78	+12.51	+12.79	+12.57
	+15.44	+5.52	+6.51	+4.48	+7.37	+7.44	+14.03	+9.17	+10.69

TABLE 7.8 – Score ROUGE (en haut) et taux de copie (en bas) sur des fractions de `valid` trié par dual_xent. Les valeurs sont exprimées par rapport aux résultats non filtrés (*i.e.* **100%**).

	BWS & R1		**BWS & Copy%**		**R1 & Copy%**	
	r	ρ	r	ρ	r	ρ
100%	43.82	28.57	69.13	83.33	13.27	9.52
95%	+1.62	0.00	+0.10	0.00	+1.36	0.00
90%	+1.30	-4.76	+0.24	0.00	+0.91	-2.38
75%	+2.72	0.00	+1.32	0.00	+3.01	0.00
50%	+9.01	+23.81	+0.22	0.00	+15.16	+23.81
25%	+21.82	+28.57	+2.41	-4.76	+32.05	+23.81
10%	+25.84	+45.24	+4.47	-4.76	+48.34	+50.00

TABLE 7.9 – Correlation entre jugement humain (BWS), ROUGE (R1) et copie (copy%) sur des fractions de `valid` trié par dual_xent. Les valeurs sont exprimées par rapport aux résultats non filtrés (*i.e.* **100%**).

7.7 Conclusion

Nous avons conduit, dans ce chapitre, une évaluation humaine de la qualité des résumés automatiques générés par nos modèles. Nous avons proposé une méthodologie d'évaluation basés sur trois critères (Grammaire, Information et Formulation). La tâche des évaluateurs consiste à déterminer le meilleur et le moins bon résumé, pour chaque critère, parmi une sélection (aléatoire) de 4 résumés issus de modèles automatiques ainsi que des résumés de référence (*i.e.* des comptes rendus de rédacteurs). En particulier, nous avons soumis à cette évaluation les 4 modèles issus du chapitre 5 ainsi que 4 modèles avec tokens spéciaux (chapitre 6) en mode prédiction, en mode informé, avec $C = 10$ et $C = 20$ tokens.

D'une part, l'évaluation des modèles est en accord avec l'évaluation automatique du chapitre 5 et appuie ses conclusions : (i) les deux approches (SELFSUP et BACKSUM) sont sensiblement meilleures que le modèle de base et (ii) la combinaison des deux approches au sein du modèle BOTH est encore meilleure.

D'autre part, nous obtenons une inadéquation entre les résultats (pourtant prometteurs) de l'évaluation automatique du chapitre 6 et le jugement humain. De plus nous observons, sur l'ensemble des modèles des résultats très similaires sur les trois critères. Cet aspect de l'évaluation, censé nous informer davantage sur les différences qualitatives s'avère peu efficace en pratique. Les scores, quels que soit le critère, semblent refléter la qualité globale du résumé davantage que le seul aspect mesuré par ce critère. Nous supposons que les modèles avec tokens, en essayant de minimiser leur extractivité s'exposent

davantage. En d'autres termes, en cherchant à s'éloigner de la source, ceux-ci ont plus de risque de générer des erreurs qui seront pénalisées par l'évaluation humaine. Une analyse plus rigoureuse des erreurs au sein des résumés permettrait de mieux comprendre les modèles, leurs qualités et leurs faiblesses.

L'évaluation humaine nous permet également de porter un regard critique sur notre cadre expérimental. D'abord, nous mesurons les corrélations entre les métriques automatiques utilisées jusqu'alors (ROUGE et le taux de copie) avec le jugement humain. Nous obtenons alors des corrélations très différentes suivant que la mesure est effectuée sur les résumés générés par les modèles sans tokens ou avec. De plus, la corrélation entre ROUGE et le jugement humain est faible : 0.44 (Pearson) et 0.29 (Spearman). ROUGE semble donc assez peu robuste dans ce contexte. Nous proposons par conséquent deux autres métriques la perplexité et la *dual cross-entropy* de [Junczys-Dowmunt, 2018]. Nous observons des corrélations plus élevées de ces métriques avec le jugement par rapport à ROUGE, en particulier la *dual cross-entropy* atteint 0.76 (Pearson) et 0.79 (Spearman) sur l'ensemble des modèles.

Enfin, nous avons considéré le filtrage des données de références (l'ensemble `valid`) par *dual cross-entropy*. La motivation était d'évaluer si les résultats obtenus sont les mêmes tandis que l'on filtre les données d'évaluation. Les modèles obtiennent de meilleurs performances à mesure que l'on filtre les données. Ce résultat peut s'expliquer par le fait que le filtre réduise le bruit au sein des données d'entrée, ce qui facilite en quelques sortes la tâche du modèle. L'ordre des modèles est globalement conservé, à part SelfSup qui obtient un meilleur score ROUGE que BackSum sur les 10% les meilleurs (alors que ce modèle est de 2 R1 inférieur sur le jeu de données complet).

Conclusion et Perspectives

Conclusion

Le travail de thèse présenté dans ce manuscrit a consisté à appliquer les approches neuronales de type apprentissage profond au domaine du résumé abstractif de transcription de parole dans le cadre de la génération de comptes rendus de réunion.

L'usage de réseaux de neurones a été motivé pour cette thèse par les récents progrès dans le domaine de la traduction neuronale et les débuts du résumé abstractif neuronal. Notre travail se distingue de l'état de l'art en cela que nous travaillons sur le français – alors que la majorité de la littérature du domaine est en anglais – et que nous traitons de comptes rendus de réunions. Nous avons identifié plusieurs verrous, en particulier (i) l'absence de jeux de données de comptes rendus en français ; (ii) la difficulté à obtenir des modèles abstractifs ; (iii) la difficulté de l'évaluation des comptes rendus obtenus du fait d'un décalage entre les métriques automatiques et le jugement humain et d'un double objectif contradictoire : maximiser ROUGE en minimisant le taux de copie [4].

Nous avons donc logiquement commencé cette thèse en nous concentrant sur la constitution d'un jeu de données pour l'entraînement supervisé de modèles de comptes rendus automatiques de réunions. Pour ce faire, nous avons proposé dans le chapitre 4 une méthodologie pour la segmentation et l'alignement des comptes rendus et des transcriptions de réunions. Cette méthodologie procède par itération, en faisant se succéder alignement humain et évaluation de modèles d'alignement automatique. Nous parvenons ainsi à faciliter la tâche d'annotation au fur et à mesure, tout en affinant nos évaluations grâce aux données de référence ainsi annotées. L'efficacité des modèles d'alignement automatique obtenus se traduit d'une part par la réduction du taux de corrections apportées par les annotateurs ainsi que par le gain substantiel des modèles de résumé automatique entraînés sur l'ensemble des données – alignées manuellement et automatiquement – par rapport aux modèles disposant uniquement des données manuelles. Ce travail à donné lieu à la publication d'un article de conférence [Tardy *et al.*, 2020a] et d'un jeu de données de test issu de réunions publiques : `public_meetings`. L'algorithme d'alignement auto-

4. Au sens strict, on ne souhaite pas minimiser le taux de copie mais le faire tendre au taux de copie de référence : celui présent entre les documents sources et les résumés cibles de l'ensemble de d'évaluation considéré.

matique utilise des vecteurs de mots pré-entraînés et nous avons utilisé seulement 138 documents pour le réglage des hyper-paramètres ce qui le rend facilement adaptable à d'autres langues ou cadres d'application.

Nous avons ensuite cherché à nous libérer des contraintes de l'alignement qui, même automatique, nécessite d'obtenir des paires des fichiers *(audio ; compte rendu)* et de calculer la transcription automatique. Dans ce but, nous avons proposé au chapitre 5 deux approches afin de tirer profit de la grande quantité de comptes rendus dont Ubiqus dispose [Tardy *et al.*, 2020b]. La première, inspirée par les récents succès de l'apprentissage auto-supervisé (notamment Devlin *et al.*, 2019 ; Radford *et al.*, 2019 ; Lewis *et al.*, 2020) consiste à pré-entraîner les modèles sur l'ensemble des comptes rendus non-alignés. Le pré-entraînement consiste alors pour le modèle à reconstituer un compte rendu à partir de sa version bruitée (mots masqués, phrases permutées, d'après Lewis *et al.*, 2020). Pour la seconde approche, nous présentons le *résumé inverse*, c'est-à-dire l'apprentissage de la génération de la source (la transcription) à partir du résumé (le compte rendu). Nous utilisons ensuite ces modèles de *résumé inverse* pour générer des transcriptions synthétiques et ainsi aligner artificiellement nos comptes rendus initialement non alignés. Ces deux approches se montrent bénéfiques pour la génération de comptes rendus en obtenant de biens meilleurs résultats que les modèles précédents. De plus, nous montrons la complémentarité des deux approches en obtenant de meilleurs résultats encore grâce à l'association du pré-entraînement auto-supervisé et de l'usage de données synthétiques lors de l'entraînement. Ayant constaté au début de la thèse la difficulté d'obtenir des données et de les aligner, nous pensons que ces approches peuvent être de puissants leviers pour les tâches souffrant du manque de ressources d'entraînement.

Les gains en performance de nos modèles exprimés par le score ROUGE sont toutefois accompagnés d'une hausse du *taux de copie*. En d'autres termes, les meilleurs modèles ont tendances à utiliser de nombreux mots issus du texte d'origine dans leurs résumés. Si l'usage de mots issus de la source n'est pas un problème en soi, nous veillons à ce que le taux de copie au sein des résumés générés reste proche du taux de copie constaté au sein des données de références (des comptes rendus de rédacteurs humains), au risque d'avoir un comportement trop *extractif*. Or, un écart significatif est constaté, dans les résumés générés par nos modèles ainsi que dans la littérature : c'est le *biais d'extractivité*. Dans le chapitre 6, nous avons donc proposé d'étudier la prédiction et le contrôle de ce paramètre grâce à des tokens spéciaux. Nous avons d'abord étudié la capacité des modèles à prédire le taux de copie attendu au sein du résumé cible, puis, l'apprentissage

conjoint de la prédiction du taux de copie et de la génération du résumé. Les résultats montrent que les modèles parviennent à prédire le taux de copie avec une plus grande exactitude que ce qui est constaté au sein des résumés obtenus précédemment. De plus, lors de l'entraînement simultané de la prédiction du taux de copie et de la génération du résumé, on observe une meilleure adéquation entre le taux de copie du résumé généré et le taux de copie attendu montrant l'efficacité de cette approche. En d'autre termes, cet apprentissage permet aux modèles de mieux adapter le taux de copie lors de leur génération. De plus, cette approche donne davantage de contrôle lors de la génération ce qui ouvre la voie à différentes évaluations en fonction du mode de génération : (i) le mode prédiction pour la comparaison avec modèles sans tokens ; (ii) le mode informé avec valeur de référence permet d'évaluer le gain de performances obtenus grâce aux informations des tokens ; (iii) on peut informer le modèle avec des valeurs quelconques afin d'évaluer les capacités d'adaptations du modèle.

Enfin, nous avons mené au sein du chapitre 7 une évaluation humaine de la qualité des comptes rendus automatiques. Nous avons ainsi évalué les comptes rendus issus de nos meilleurs modèles face à une référence humaine sur trois critères : *Grammaire*, *Information* et *Formulation*. Si les résultats de cette évaluation sont en adéquation avec ceux du chapitre 5, ils semblent contredire les conclusions du chapitre 6 : les modèles avec tokens obtiennent un jugement humain moins favorable que leurs équivalents sans tokens. Ce résultats est particulièrement étonnant pour une approche qui semblait prometteuse, en parvenant à sensiblement abaisser le taux de copie tout en conservant de bonnes performances en termes de score ROUGE. Cette invalidation par l'évaluation humaine peut être un indicateur quant à la difficulté à obtenir un modèle plus abstractif. En effet, réduire la copie au sein d'un modèle le force à s'éloigner du texte source ce qui le rend plus prompt à commettre des erreurs. On observe par exemple davantage de mots nouveaux absents de la cible, c'est-à-dire présents ni dans la source, ni dans la cible (*cf.* mots en bleu dans les extraits de l'Annexe B) . Ces mots nouveaux montrent la capacité d'abstractivité de ces modèles mais peuvent être erroné, ou à minima ne participent à au score ROUGE.

Enfin, nous rapportons dans le tableau 7.10 les résultats en termes de score ROUGE, de taux de copie et de jugement humain obtenus par les différents modèles développés au cours de cette thèse.

	valid			public_meetings	
	Score ROUGE	Copy%	BWS	Score ROUGE	Copy%
man	30.29 / 11.19 / 27.32	**60.75**	–	44.03 / 23.74 / 41.09	69.18
man+auto	33.83 / 15.86 / 31.05	74.25	-0.32	52.31 / 34.00 / 49.70	79.36
man (L)	–	–	–	52.80 / 29.59 / 49.49	73.09
man+auto (L)	–	–	–	56.56 / 35.43 / 53.55	**77.62**
SelfSup	37.94 / 19.16 / 34.86	78.62	**0.06**	55.08 / 36.76 / 52.43	83.72
BackSum	39.17 / 20.06 / 36.30	86.96	**0.10**	55.29 / 36.39 / 52.96	89.24
Both	**40.23 / 21.02 / 37.26**	88.03	**0.26**	**58.02 / 38.64 / 55.56**	90.77
Pred10	34.60 / 15.18 / 31.46	**64.47**	-0.17	51.01 / 32.75 / 48.42	71.78
Pred20	36.06 / 17.11 / 33.32	68.97	-0.02	54.44 / 35.28 / 51.99	80.64
Inf10	38.40 / 16.69 / 34.92	**59.96**	-0.36	56.11 / 36.20 / 53.28	**73.43**
Inf20	**40.75 / 19.18 / 37.39**	66.06	-0.03	**58.39 / 37.50 / 55.61**	**77.99**
PredBoth	**40.43 / 20.45 / 37.48**	78.32	–	**56.58 / 36.88 / 54.03**	85.62

TABLE 7.10 – Score ROUGE, taux de copie et jugement humain (*BWS*, score dans $[-1; 1]$) des différents modèles développés durant cette thèse sur deux jeux d'évaluation : `valid` et `public_meetings`. Les modèles *man* et *man+auto* proviennent du chapitre 4 (le *L* signifie *avec information de longueur*) ; *SelfSup*, *BackSum* et *Both* ont été développé dans le chapitre 5 ; *Pred10*, *Inf10*, *Pred20*, *Inf20* et *PredBoth* désignent les modèles avec tokens du chapitre 6 en mode prédiction (*Pred*) ou informé de la longueur et du taux de copie (*Inf*). Les 3 meilleures valeurs de chaque colonnes sont en gras. Les taux de copie de référence sont de 55.38% pour `valid` et 75.84% pour `public_meetings`.

Perspectives

Au cours de cette thèse, nous avons d'abord rencontré des problématiques liées à la disponibilité des données (chapitres 4 et 5) avant d'être confronté à des questions d'évaluation. En effet, afin de progresser sur notre tâche, il a été nécessaire de disposer d'un cadre expérimental permettant de mesurer quantitativement les avancées. Dans notre cas, et comme il est d'usage dans la littérature scientifique, nous avons constitué des jeux de données d'évaluation et utilisé une métrique automatique : ROUGE. Dès l'observation du *biais d'extractivité*[5], nous avons décider de mesurer également le taux de copie de nos modèles. Nous avons ainsi constaté un paradoxe dans nos métriques : l'augmentation du score ROUGE censée traduire une meilleure performances, s'accompagne d'une augmentation du taux de copie, ce qui amoindrit le caractère abstractif des modèles. Il est alors difficile de mesurer les progrès effectués et de déterminer les directions à prendre alors que deux de nos objectifs semblent s'opposer.

Il nous semble donc qu'il serait très bénéfique pour le domaine du résumé abstractif de développer des métriques plus spécifiques. En effet, en n'utilisant que ROUGE pour seule mesure de la qualité d'un résumé nous manquons de finesse d'analyse. Il serait par exemple intéressant de distinguer les capacités grammaticales des modèles, leur exactitude factuelle (c'est-à-dire le fait de générer des résumés qui soit factuellement en accord avec la source) ou leur capacité de reformulation. Un début de réponse à cette problématique est apporté par le développement de nouvelles métriques neuronales d'évaluation des résumés automatiques [Eyal *et al.*, 2019 ; T. Zhang *et al.*, 2019]. D'autre part, [Fabbri *et al.*, 2021] présentent une large ré-évaluation de modèles, de jeux de données et de métriques d'évaluation. Ils proposent notamment de comparer les corrélations des métriques d'évaluations automatique par rapport au jugement humain sur 4 critères (cohérence, exactitude factuelle, orthographe, pertinence), et ce à partir de résumés automatique et humains.

Par conséquent, en raison des faiblesses de l'évaluation automatique, les chercheurs procèdent généralement à une évaluation humaine afin de confirmer leurs résultats. Celle-ci présente pourtant également des difficultés.

— **Faible accord inter-annotateur.** Tout d'abord, plusieurs travaux récents pointent la difficulté d'obtenir un accord inter-annotateur satisfaisant [D. M. Ziegler *et al.*, 2019 ; Fabbri *et al.*, 2021]. En particulier, [Stiennon *et al.*, 2020] mettent en avant

5. Le *biais d'extractivité* désigne la tendance des les modèles de résumé automatique à trop copier des mots de la source dans leur résumé.

la différence de jugement entre les évaluateurs issus de plateformes collaboratives et les évaluateurs experts. Ils soulignent alors l'importance de la communication afin d'établir des règles communes entre les évaluateurs.

— **Critères d'évaluation peu discriminants.** De plus, malgré les efforts des chercheurs d'intégrer différents critères à leurs évaluations humaines, les résultats montrent souvent une grande inter-corrélation de ces critères. En d'autres termes, la qualité globale du résultat biaise chacun des critères si bien qu'ils finissent par mesurer la même chose (c'est notamment un des points observé lors de notre évaluation, chapitre 7 ainsi que par Fabbri *et al.*, 2021).

— **Biais humain d'extractivité** Enfin, de même que nous discuté du *biais d'extractivité* au sein des modèles de résumé automatique (*cf.* section 3.6 et chapitre 6), les modèles extractifs peuvent être favorisés lors de l'évaluation humaine. [Stiennon *et al.*, 2020] met en garde contre ce phénomène qu'ils parviennent à limiter grâce à la formation des évaluateurs humains.

Pour toute ces raisons, nous pensons que l'établissement d'un protocole commun d'évaluation humaine est une perspective intéressante pour l'avancée du domaine.

Dans notre cas précis, il serait pertinent d'approfondir l'évaluation humaine et de l'étendre à davantage de paires d'évaluation et de modèles. En particulier, des modèles obtenant de très bons résultats lors de l'évaluation automatique (*e.g.* `D-BOTH`) n'ont pas été soumis à l'évaluation humaine. De plus, afin de mieux comprendre les différences entre les modèles il serait pertinent de mener une analyse rigoureuse des erreurs afin de déterminer précisément les difficultés que rencontrent les modèles.

Un autre axe de recherche concerne le développement de modèles plus abstractifs. Nous avons pu constater la difficulté de cette tâche qui peut s'accompagner d'une baisse de performances en termes de ROUGE ainsi qu'en termes de jugement humain. En effet, les approches extractives ont en général l'avantage de fournir des phrases plus correctes, celles-ci étant issues du document source. À l'inverse, chercher à accroître la diversité au sein des résumés expose les modèles à davantage de risques. L'approche extractive s'apparente alors à un optimum local dont il peut être difficile de s'extraire sans sacrifier – localement – en performance. Toutefois, des travaux intéressants s'engagent sur la voie de l'augmentation de l'abstractivité, que ce soit grâce à de l'apprentissage par renforcement avec fonctions de récompenses favorisant la génération de nouveaux mots [Kryściński *et al.*, 2020] ou avec des stratégies de prédictions favorisant la diversité [Scialom *et al.*, 2020c].

Bibliographie

Bahdanau, D., Cho, K., Bengio, Y., Dzmitry Bahdana, Bahdanau, D., Cho, K. & Bengio, Y., (2014), Neural Machine Translation By Jointly Learning To Align and Translate, *Iclr 2015*, 1-15, https://doi.org/10.1146/annurev.neuro.26.041002.131047

Beeferman, D., Berger, A., Lafferty, J., Cardie, C. & Mooney, R., (1999), *Statistical Models for Text Segmentation* (rapp. tech. N° 1), Kluwer Academic Publishers, https://doi.org/10.1023/A:1007506220214

Bengio, Y., Ducharme, R., Vincent, P., Janvin, C., Jauvin, C., Ca, J. U., Kandola, J., Hofmann, T., Poggio, T. & Shawe-Taylor, J., (2000), 10.1162/153244303322533223, *CrossRef Listing of Deleted DOIs*, *1*, 1137-1155, https://doi.org/10.1162/153244303322533223

Bengio, Y., Simard, P. & Frasconi, P., (1994), Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks*, *5 2*, 157-166, https://doi.org/10.1109/72.279181

Böhm, F., Gao, Y., Meyer, C. M., Shapira, O., Dagan, I. & Gurevych, I., (2019), Better rewards yield better summaries : Learning to summarise without references, *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3110-3120, https://doi.org/10.18653/v1/d19-1307

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D., (2020), Language Models are Few-Shot Learners, *Advances in Neural Information Processing Systems*, *33*, https://commoncrawl.org/the-data/

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y., (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1724-1734, https://doi.org/10.3115/v1/d14-1179

Chopra, S., Auli, M., Rush, A. M. & Seas, H., (2016), Abstractive Sentence Summarization with Attentive Recurrent Neural Networks, *NAACL-2016*, http://nlp.seas.harvard.edu/papers/naacl16_summary.pdf

Choi, F. Y. Y., (2000), Advances in domain independent linear text segmentation, *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, https://www.aclweb.org/anthology/A00-2004

Collobert, R. & Weston, J., (2008), A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning, *Proceedings of the 25th International Conference on Machine Learning*, 160-167, https://doi.org/10.1145/1390156.1390177

Conneau, A., Kiela, D., Schwenk, H., Barrault, L. & Bordes, A., (2017), Supervised learning of universal sentence representations from natural language inference data, *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 670-680, https://doi.org/10.18653/v1/d17-1070

Daumé, H. & Marcu, D., (2009), A Noisy-Channel Model for Document Compression, *Journal of the American Society for Information Science*, *416*, 391-407, https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Denkowski, M. & Lavie, A., (2015), Meteor Universal : Language Specific Translation Evaluation for Any Target Language, *Proceedings of the ninth workshop on statistical machine translation*, 376-380, https://doi.org/10.3115/v1/w14-3348

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., (2019), {BERT} : Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171-4186, https://doi.org/10.18653/v1/N19-1423

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M. & Hon, H. W., (2019), Unified Language Model Pre-training for Natural Language Understanding and Generation, *33rd Conference on Neural Information Processing Systems*, https://github.com/microsoft/unilm

Duchi, J., Hazan, E. & Singer, Y., (2010), *Adaptive subgradient methods for online learning and stochastic optimization* (rapp. tech.).

Edunov, S., Ott, M., Auli, M., Grangier, D., Park, M., Brain, G. & View, M., (2018), Understanding Back-Translation at Scale, *Proceedings of the 2018 Confe-*

rence on Empirical Methods in Natural Language Processing, EMNLP 2018, 489-500, https://doi.org/10.18653/v1/d18-1045

Eyal, M., Baumel, T. & Elhadad, M., (2019), Question answering as an automatic evaluation metric for news article summarization, *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies - Proceedings of the Conference*, *1*, 3938-3948, https://doi.org/10.18653/v1/n19-1395

Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R. & Radev, D., (2021), SummEval : Re-evaluating Summarization Evaluation, *Transactions of the Association for Computational Linguistics*, *9*, 391-409, https://doi.org/10.1162/tacl{_}a{_}00373

Fan, A., Grangier, D. & Auli, M., (2018), Controllable abstractive summarization, *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, 45-54, https://doi.org/10.18653/v1/w18-2706

Fauconnier, J.-P., (2015), French Word Embeddings, http://fauconnier.github.io

Firth, J. R., (1957), Papers in Linguistics, *International Journal of Applied Linguistics*, *173*, 402-413, https://doi.org/10.1111/j.1473-4192.2007.00164.x

Gehrmann, S., Deng, Y. & Rush, A. M., (2018), Bottom-Up Abstractive Summarization, *Proceedings of EMNLP*, http://arxiv.org/abs/1808.10792

Graff, D., Kong, J., Chen, K. & Maeda, K., (2003), English gigaword, *Linguistic Data Consortium, Philadelphia*, *4 1*, 34.

Graves, A., (2013), Generating Sequences With Recurrent Neural Networks, http://arxiv.org/abs/1308.0850

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B. & Bengio, Y., (2016), Pointing the Unknown Words, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 140-149, https://doi.org/10.18653/v1/P16-1014

Heafield, K., (2011), KenLM : Faster and Smaller Language Model Queries, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 187-197.

Hearst, M. A., (1997), TextTiling : Segmenting text into multi-paragraph subtopic passages, *Computational Linguistics*, *23 1*, 33-64, http://www.aclweb.org/anthology/J97-1003

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. & Blunsom, P., (2015), Teaching Machines to Read and Comprehend, *Ad-*

vances in Neural Information Processing Systems, *2015-Janua*, 1693-1701, http://arxiv.org/abs/1506.03340

Hernandez, F., Nguyen, V., Ghannay, S., Tomashenko, N. & Estève, Y., (2018), TED-LIUM 3 : Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11096 LNAI*, 198-208, https://doi.org/10.1007/978-3-319-99579-3

Hinton, G., Srivastava, N. & Swersky, K., (2012), Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, *Cited on*, *14 8*.

Hochreiter, S. & Schmidhuber, J., (1997), Long Short-Term Memory, *Neural Computation*, *9 8*, 1735-1780, https://doi.org/10.1162/neco.1997.9.8.1735

Junczys-Dowmunt, M., (2018), Dual Conditional Cross-Entropy Filtering of Noisy Parallel Corpora, *Proceedings of the Third Conference on Machine Translation : Shared Task Papers*, *2*, 888-895, https://doi.org/10.18653/v1/W18-6478

Katz, S. M., (1987), *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer* (rapp. tech. N° 3).

Kingma, D. P. & Ba, J. L., (2015), Adam : A method for stochastic optimization, *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, https://arxiv.org/abs/1412.6980v9

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R. & Fidler, S., (2015), Skip-thought vectors, *Advances in Neural Information Processing Systems*, *2015-Janua*, 3294-3302.

Kiritchenko, S. & Mohammad, S., (2017), Best-Worst Scaling More Reliable than Rating Scales : A Case Study on Sentiment Intensity Annotation, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, *2*, 465-470, https://doi.org/10.18653/v1/P17-2074

Klein, G., Kim, Y., Deng, Y., Senellart, J. & Rush, A. M., (2017), OpenNMT : Open-source toolkit for neural machine translation, *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, 67-72, https://doi.org/10.18653/v1/P17-4012

Kryściński, W., Paulus, R., Xiong, C. & Socher, R., (2020), Improving abstraction in text summarization, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 1808-1817, https://doi.org/10.18653/v1/d18-1207

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L., (2020), *BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* (rapp. tech.), https://doi.org/10.18653/v1/2020.acl-main.703

Lin, C. Y., (2004), Rouge : A package for automatic evaluation of summaries, *Proceedings of the workshop on text summarization branches out (WAS 2004)*, 25-26, http://www.aclweb.org/anthology/W04-1013

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, Ł., Shazeer, N., Kaiser, L., Shazeer, N., J. Liu, P., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L. & Shazeer, N., (2018), Generating Wikipedia by Summarizing Long Sequences, *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, http://arxiv.org/abs/1801.10198

Liu, Y. & Lapata, M., (2020), Text summarization with pretrained encoders, *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3730-3740, https://doi.org/10.18653/v1/d19-1387

Lund, K. & Burgess, C., (1996), Producing high-dimensional semantic spaces from lexical co-occurrence, *Behavior Research Methods, Instruments, and Computers*, *28 2*, 203-208, https://doi.org/10.3758/BF03204766

Luong, T., Sutskever, I., Le, Q., Vinyals, O. & Zaremba, W., (2015), Addressing the Rare Word Problem in Neural Machine Translation, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, 11-19, http://arxiv.org/abs/1410.8206

Mathur, N., Baldwin, T. & Cohn, T., (2020), Tangled up in BLEU : Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics, 4984-4997, https://doi.org/10.18653/v1/2020.acl-main.448

Mcculloch, W. S. & Pitts, W., (1943), A logical calculus of the ideas immanent in nervous activity, *The Bulletin of Mathematical Biophysics*, *5 4*, 115-133, https://doi.org/10.1007/BF02478259

Mikolov, T., Karafiát, M., Burget, L., Jan, C. & Khudanpur, S., (2010), Recurrent neural network based language model, *Proceedings of the 11th Annual Confe-*

rence of the International Speech Communication Association, INTERSPEECH 2010, 1045-1048.

Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, I., Chen, K., Corrado, G. & Dean, J., (2013a), Distributed Representations of Words and Phrases and their Compositionality, *Nips*, https://doi.org/10.1162/jmlr.2003.3.4-5.951

Mikolov, T., Yih, W. T. & Zweig, G., (2013b), Linguistic regularities in continuous spaceword representations, *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Proceedings of the Main Conference*, 746-751, https://www.aclweb.org/anthology/N13-1090

Mikolov, T., Chen, K., Corrado, G. & Dean, J., (2013c), Efficient estimation of word representations in vector space, *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, *1*, 1-12, https://doi.org/10.1162/153244303322533223

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç. & Xiang, B., (2016), Abstractive text summarization using sequence-to-sequence RNNs and beyond, *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, 280-290, https://doi.org/10.18653/v1/k16-1028

Narayan, S., Cohen, S. B. & Lapata, M., (2020), Don't give me the details, just the summary ! Topic-aware convolutional neural networks for extreme summarization, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 1797-1807, https://doi.org/10.18653/v1/d18-1206

Over, P., Dang, H. & Harman, D., (2007), DUC in context, *Information Processing & Management*, *436*, 1506-1520, https://doi.org/10.1016/j.ipm.2007.01.019

Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J., (2002), BLEU : a method for automatic evaluation of machine translation, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 311-318, https://doi.org/10.3115/1073083.1073135

Paulus, R., Xiong, C. & Socher, R., (2017), A Deep Reinforced Model for Abstractive Summarization, *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, http://arxiv.org/abs/1705.04304

Pennington, J., Socher, R. & Manning, C. D., (2014), GloVe : Global vectors for word representation, *EMNLP 2014 - 2014 Conference on Empirical Methods in*

Natural Language Processing, Proceedings of the Conference, 1532-1543, https://doi.org/10.3115/v1/d14-1162

Pevzner, L. & Hearst, M. A., (2002), A critique and improvement of an evaluation metric for text segmentation, *Computational Linguistics*, *28 1*, 19-36, https://doi.org/10.1162/089120102317341756

Poncelas, A., Shterionov, D., Way, A., De Buy Wenniger, G. M. & Passban, P., (2018), Investigating backtranslation in neural machine translation, *EAMT 2018 - Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, 249-258.

Povey, D., Glembek, O., Goel, N., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovsk´y, J. S., Stemmer, G. & Vesel´y, K. V., (2011), *The Kaldi speech recognition toolkit Speech Recognition View project IARPA Babel View project The Kaldi Speech Recognition Toolkit* (rapp. tech.), http://kaldi.sf.net/

Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohamadi, M. & Khudanpur, S., (2018), Semi-orthogonal low-rank matrix factorization for deep neural networks, *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, *2018-Septe*, 3743-3747, https://doi.org/10.21437/Interspeech.2018-1417

Radford, A., Salimans, T., Narasimhan, K., Salimans, T. & Sutskever, I., (2018), Improving Language Understanding by Generative Pre-Training, *OpenAI*, 1-12.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I., (2019), Language Models are Unsupervised Multitask Learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J., (2020), Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *Journal of Machine Learning Research*, *21*, 1-67, http://jmlr.org/papers/v21/20-074.html.

Ranzato, M. M., Chopra, S., Auli, M. & Zaremba, W., (2016), Sequence level training with recurrent neural networks, *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, https://github.com/facebookresearch/MIXER

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J. & Goel, V., (2017), Self-critical sequence training for image captioning, *Proceedings - 30th IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua, 1179-1195, https://doi.org/10.1109/CVPR.2017.131

Rosenblatt, F., (1958), The perceptron : A probabilistic model for information storage and organization in the brain, *Psychological Review, 65 6*, 386-408, https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E. & Williams, R. J., (1986), Learning representations by back-propagating errors, *Nature, 323 6088*, 533-536, https://doi.org/10.1038/323533a0

Rush, A. M., Chopra, S. & Weston, J., (2015), A Neural Attention Model for Abstractive Sentence Summarization, *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 379-389, https://doi.org/10.1162/153244303322533223

Sak, H., Senior, A. & Beaufays, F., (2014), Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition, http://arxiv.org/abs/1402.1128

Sankaran, B., Mi, H., Al-Onaizan, Y. & Ittycheriah, A., (2016), Temporal Attention Model for Neural Machine Translation, *Arxiv, 2014*, http://arxiv.org/abs/1608.02927

Sandhaus, E., (2008), The new york times annotated corpus, *Linguistic Data Consortium, Philadelphia*, https://doi.org/https://doi.org/10.35111/77ba-9x74

Schwenk, H. & Gauvain, J. L., (2005), Training neural network language models on very large corpora, *HLT/EMNLP 2005 - Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 201-208, https://doi.org/10.3115/1220575.1220601

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B. & Staiano, J., (2020a), MLSUM : The Multilingual Summarization Corpus, https://doi.org/10.18653/v1/2020.emnlp-main.647

Scialom, T., Lamprier, S., Piwowarski, B. & Staiano, J., (2020b), Answers unite! Unsupervised metrics for reinforced summarization models, *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3246-3256, https://doi.org/10.18653/v1/d19-1320

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B. & Staiano, J., (2020c), ColdGANs : Taming Language GANs with Cautious Sampling Strategies, http://arxiv.org/abs/2006.04643

See, A., Liu, P. J. & Manning, C. D., (2017), Get to the point : Summarization with pointer-generator networks, *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*, 1073-1083, https://doi.org/10.18653/v1/P17-1099

Sennrich, R., Haddow, B. & Birch, A., (2016), Improving Neural Machine Translation Models with Monolingual Data, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, *1*, 86-96, https://doi.org/10.18653/v1/p16-1009

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D. & Christiano, P., (2020), Learning to summarize from human feedback, http://arxiv.org/abs/2009.01325

Sutskever, I., Martens, J., E. Hinton, G. & Hinton, G., (2011), Generating Text with Recurrent Neural Networks, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017-1024, http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf

Sutskever, I., Vinyals, O. & Le, Q. V., (2014), Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems*, *4 January*, 3104-3112, http://arxiv.org/abs/1409.3215

Tardy, P., Janiszek, D., Estève, Y. & Nguyen, V., (2020a), Align then Summarize : Automatic Alignment Methods for Summarization Corpus Creation, *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 6718-6724, https://github.com/pltrdy/autoalign

Tardy, P., de Seynes, L., Hernandez, F., Nguyen, V., Janiszek, D. & Estève, Y., (2020b), Leverage unlabeled data for abstractive speech summarization with self-supervised learning and back-summarization, *International Conference on Speech and Computer (SPECOM 2020)*, 572-580, https://doi.org/10.1007/978-3-030-60276-5{_}55

Tu, Z., Lu, Z., Liu, Y., Liu, X. & Li, H., (2016), Modeling Coverage for Neural Machine Translation, *Proceedings of the 21st International Conference on Intelligent User Interfaces - IUI '16*, 228-240, https://arxiv.org/abs/1601.04811

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I., Kaiser, L. & Polosukhin, I., (2017), Attention is All you Need, In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Éd.), *Advances in Neural Information Processing Systems 30* (p. 5998-6008), Curran Associates, Inc., https://doi.org/10.1017/S0140525X16001837

Vedantam, R., Zitnick, C. L. & Parikh, D., (2015), CIDEr : Consensus-based image description evaluation, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 4566-4575, https://doi.org/10.1109/CVPR.2015.7299087

Vinyals, O., Fortunato, M. & Jaitly, N., (2015), Pointer Networks (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama & R. Garnett, Éd.), *Advances in Neural Information Processing Systems 28*, 2692-2700, http://arxiv.org/abs/1506.03134

Williams, R. J., (1992), Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning*, *8 3*, 229-256, https://doi.org/10.1023/A:1022672621406

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. & Le, Q. V., (2019), XLNet : Generalized Autoregressive Pretraining for Language Understanding, In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox & R. Garnett (Éd.), *Advances in Neural Information Processing Systems* (p. 5753-5763), Curran Associates, Inc., https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q. & Artzi, Y., (2019), BERTScore : Evaluating Text Generation with BERT, *arXiv*, http://arxiv.org/abs/1904.09675

Zhang, J., Zhao, Y., Saleh, M. & Liu, P. J., (2020), PEGASUS : Pre-Training with extracted gap-sentences for abstractive summarization.

Ziegler, Z. M., Melas-Kyriazi, L., Gehrmann, S. & Rush, A. M., (2019), Encoder-Agnostic Adaptation for Conditional Language Generation, *CoRR*, *abs/1908.0*, http://arxiv.org/abs/1908.06938

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P. & Irving, G., (2019), Fine-Tuning Language Models from Human Preferences, http://arxiv.org/abs/1909.08593

Annexes

Annexe A

Contributions Logicielles

— `OpenNMT-py` [1] : *framework* initialement crée pour la traduction neuronale (= *NMT*) avec des modèles encodeur-décodeur à base de RNN, `OpenNMT-py` implémente maintenant le modèle Transformer et est appliqué aux autres tâches de génération de texte, dont le résumé automatique. Le projet a été initié par Systran et Harvard-NLP avant d'être essentiellement maintenu par Ubiqus et la communauté.

— `autoalign` [2] : implémentation de l'alignement automatique (chapitre 4) et code pour reproduction de [Tardy *et al.*, 2020a].

— `public_meetings` [3] : jeu de données de test consistant en 1029 paires de *(transcription ; compte rendu)* issues de réunion publiques traitées par Ubiqus. Ce corpus a été publié dans [Tardy *et al.*, 2020a].

— `files2rouge` [4] : un *wrapper* en Python autour de l'implémentation officielle du score ROUGE permettant de comparer deux fichiers contenants une séquence par ligne correspondants respectivement au résumé à évaluer et au résumé cible.

— `rouge` [5] : implémentation en Python de la métrique ROUGE. Ce projet à l'avantage d'être bien plus rapide par rapport aux *wrapper* (comme `files2rouge`).

— `Ext-Oracle` [6] : il est courant d'appeler « Oracle » une approche qui utilise des informations théoriquement inaccessible. Ici, `Ext-Oracle` est un outil pour générer des résumés extractif avec la connaissance du résumé cible. Le résumé est ainsi généré en sélectionnant des phrases de la source qui maximise le score ROUGE par rapport à la référence. Cette approche est utilisée afin d'estimer la borne supérieure en termes de score ROUGE atteignable par un modèle extractif.

— `lead_sentences` [7] : cet outil génère des résumés à partir de textes en sélectionnant

1. https://github.com/OpenNMT/OpenNMT-py
2. https://github.com/pltrdy/autoalign
3. https://github.com/pltrdy/public_meetings
4. https://github.com/pltrdy/files2rouge
5. https://github.com/pltrdy/rouge
6. https://github.com/pltrdy/extoracle_summarization
7. https://github.com/pltrdy/lead_sentences/

les premières phrases de celui-ci. La performance de cette approche – aussi naïve soit elle – est mesurée (en termes de score ROUGE) et nous informe sur une éventuelle concentration des informations importantes en début de document. Le nombre de phrases sélectionnées correspond au nombres de phrases du résumé cible.

Annexe B

Extraits de comptes rendus automatiques

Nous présentons dans cette annexe des exemples de résumés issus des modèles développés au cours de cette thèse. Les paires de textes sources (*src*) et cibles (*tgt*) sont tirés du jeu de test `public_meetings` [Tardy *et al.*, 2020a]. Les exemples ont été tirés aléatoirement avec comme seule contrainte de ne pas être trop longs et que la source et la cible soient correctement alignées (il existe quelques erreurs d'alignement au sein du jeu de données).

Nous mettons en avant différentes caractéristiques grâce à un code couleur :

— En **vert** : mots nouveaux – c'est -à-dire absent de la source – également présent dans le résumé cible. Ils représentent donc l'abstractivité des résumés.

— En **bleu** : mot nouveaux absents du résumé cible. Il s'agit aussi de démonstration de l'abstractivité du modèle, mais on ne sait pas à priori si cela est pertinent (il peut s'agir d'erreur également).

— En **jaune** : il s'agit de mots copiés de la source mais absent du résumé cible. Il peut donc s'agir d'excès de copie.

src : nous avons été portés par une activité de nos clients qui a été assez bonne , assez dynamique , c' est-à-dire que l' hôtellerie s' est très bien comportée , les hôtels que nous servons ont eu eux-mêmes une bonne activité , bien meilleur que deux mille seize . nous avons bénéficié de ce rebond de l' activité sur l' hôtellerie restauration , et cela a soutenu notre croissance dans le pays . il faut souligner aussi que nous a bien mieux maitriser la compétition prix et l' ensemble des efforts que nous avons déployé pour maîtriser notre pricing commence à porter ses fruits . nous avons été bien plus efficaces sur le sujet pris en france . tout ceci nous a permis de très légèrement améliorer notre marge en france . c' était un sujet qui était extrêmement important pour l' entreprise , puisque nous avions été en dégradation de marge sur la france en deux mille quinze et en deux mille seize . c' était très important pour nous de montrer que nous étions capables en deux mille dix-sept d' inverser cette tendance et de repartir en croissance de marge sur la france . c' est ce que nous avons réussi à faire

---

tgt : en france , l' année 2017 restera un " bon cru " pour elis en france . nous avons été portés par une activité assez bonne , assez dynamique , de nos clients , c' est-à-dire que l' hôtellerie s' est très bien comportée : l' activité des hôtels que nous servons a été bien meilleure qu' en 2016 , et nous avons bénéficié de ce rebond de l' activité sur l' hôtellerie-restauration , qui a soutenu notre croissance dans le pays . il faut également souligner que nous avons bien mieux maîtrisé la compétition prix . l' ensemble des efforts que nous avons déployés pour maîtriser notre pricing commence à porter ses fruits . nous avons donc été bien plus efficaces en matière de prix en france . tout ceci nous a donc permis de très légèrement améliorer notre marge en france . ce sujet était extrêmement important pour l' entreprise , puisque notre marge était en dégradation en france en 2015 et en 2016 . il était très important pour nous de montrer que nous étions capables en 2017 d' inverser cette tendance et de repartir en croissance de marge sur la france : c' est ce que nous avons réussi à faire .

---

man+auto : nous avons été portés par une activité de nos clients , qui a été assez bonne dynamique . en 2016 , nous avons bénéficié d' un rebond de l' hôtellerie , qui s' est comporté notre croissance dans le pays . il convient également de souligner que nous maîtrisons la compétition prix et l' ensemble des efforts que nous avons déployé pour maîtriser notre pricing , commencer à porter ses fruits . tout ceci nous a permis d' améliorer notre marge en france , ce qui constituait un sujet extrêmement important pour l' entreprise puisque nous avions été en dégradation de marge sur la france en 2015 et en 2016 .

---

selfsup : nous nous sommes également portés par une activité de nos clients . les coûts de l' hôtellerie s' y sont bien comportés , à hauteur de 3 000 tonnes . nous avons bénéficié de ce rebond de la croissance dans le pays , pour mieux maîtriser la compétition prix . l' ensemble des efforts déployés pour maîtriser notre pricing commence à porter ses fruits .

---

backsum : en 2016 , nous avons bénéficié de ce rebond de l' activité sur l' hôtellerie restauration , qui a soutenu notre croissance dans le pays . il faut souligner que nous avons bien mieux maîtriser la compétition prix . l' ensemble des efforts déployés pour maîtriser notre pricing commence à porter ses fruits . nous avons été bien plus efficaces sur le sujet pris en france , ce qui nous a permis de très légèrement améliorer notre marge en france . il s' agissait d' un sujet extrêmement important pour l' entreprise , puisque nous étions capables en 2017 d' inverser cette tendance et de repartir en croissance de marge sur la france .

---

both : nous avons été portés par une activité de nos clients qui a été assez bonne et assez dynamique . l' hôtellerie s' est très bien comportée . les hôtels que nous servons ont eu eux-mêmes une bonne activité , bien meilleur que 2016 . nous avons bénéficié de ce rebond de l' activité sur l' hébergement et cela a soutenu notre croissance dans le pays . il faut souligner aussi que nous avons bien mieux maitriser la compétition prix et l' ensemble des efforts déployés pour maîtriser notre pricing commencent à porter ses fruits . nous sommes bien plus efficaces sur le sujet pris en france . tout ceci nous a permis de très légèrement améliorer notre marge en france , ce qui était extrêmement important pour l' entreprise .

---

pred20 : nous nous sommes portés par une activité de nos clients qui s' est très bien comportée . les hôtels que nous nous servons ont eu eux-mêmes une bonne activité bien meilleur que 2016 . nous avons bénéficié de ce rebond de l' activité sur l' hôtellerie restauration et a soutenu notre croissance dans le pays . nous devons mieux maîtriser la compétition prix . l' ensemble des efforts déployés pour maîtriser notre pricing commencent à porter ses fruits .

---

inf20 : nous nous sommes portés par une activité de nos clients qui s' est avérée assez bonne . les hôtels que nous nous servons ont même eu une bonne activité bien meilleur que 2016 . nous avons bénéficié de ce rebond de l' activité sur l' hôtellerie-de-moselle et la restauration . nous devons mieux maîtriser la compétition prix . l' ensemble des efforts que nous avons déployés pour maîtriser notre pricing commencent à porter ses fruits . ceci nous a permis de très légèrement améliorer notre marge en france . nous avions été en dégradation de marge en 2015 et 2016 . il était important pour nous de montrer que nous étions capables d' inverser cette tendance et de repartir en croissance de marge sur la france . c' est l' essence de notre travail . nous espérons poursuivre cette croissance en 2017 . par ailleurs , nous avons réussi à nous montrer plus efficaces sur le sujet de la compétitivité prix .

---

predboth : nous avons été portés par une activité de nos clients assez dynamique . l' hôtellerie s' est très bien déroulée . les hôtels que nous servons ont connu eux-mêmes une bonne activité , bien meilleure que 2016 . nous avons bénéficié de ce rebond de l' activité sur la restauration et avons soutenu notre croissance dans le pays . il faut souligner que nous avons mieux maîtriser la compétition prix et l' ensemble des efforts déployés pour maîtriser notre pricing commencent à porter ses fruits . ce dernier nous a permis de très légèrement améliorer notre marge en france .

FIGURE B.1 – Extraits de comptes rendus automatiques n°1

src : pour votre entreprise ? merci beaucoup . vous avez maintenant NOM va nous présenter les résultats financiers de l' entreprise de deux mille dix-sept et du premier trimestre

---

tgt : merci beaucoup PRENOM . PRENOM NOM nous présentera maintenant les résultats financiers de l' entreprise en 2017 et sur le premier trimestre 2018 .

---

man+auto : merci pour votre entreprise . avez -vous des questions ? merci beaucoup . nous allons vous présenter les résultats financiers de l' entreprise de 2017 et du premier trimestre .

---

selfsup : vous avez maintenant la possibilité de nous présenter les résultats financiers de l' entreprise de 2017 et du premier trimestre .

---

backsum : vous avez maintenant l' intention de nous présenter les résultats financiers de l' entreprise en 2017 et du premier trimestre 2017 .

---

both : m. le president . – merci beaucoup . vous avez maintenant la présentation des résultats financiers de l' entreprise de 2017 et du premier trimestre 2017 .

---

pred20 : vous avez maintenant PRENOM balt va nous présenter les résultats financiers de l' entreprise de 2017 et du premier trimestre .

---

inf20 : vous avez maintenant monsieur PRENOM , qui va nous présenter les résultats financiers de l' entreprise de 2017 et du premier trimestre .

---

predboth : vous avez maintenant monsieur NOM qui va nous présenter les résultats financiers de l' entreprise de 2017 et du premier trimestre 2017 .

FIGURE B.2 – Extraits de comptes rendus automatiques n°2

src : sur le royaume-uni , ils en ont . il en va tout autrement puisque PRENOM a souligné les difficultés opérationnelles de la zone . on était à l' époque sur du moins quatre en tendance pro forma , reflétant bien les difficultés opérationnelles . vous avez vu avec la publication du chiffre d' affaires du q 1 qu' on est déjà plus qu' à moins de deux signe du redressement qui s' opère . l' europe du sud , donc essentiellement l' espagne et le portugal très dynamique , avec une croissance organique à plus de cinq , malgré PRENOM l' a souligné , un ralentissement de l' activité hôtelière en espagne .

---

tgt : au royaume-uni , il en va tout autrement . PRENOM a souligné les difficultés opérationnelles de la zone , qui connaissait en 2017 une décroissance de -4 % en pro forma . la publication du chiffre d' affaires du q 1 vous a montré qu' elle n' atteint déjà plus que -2 % , signe du redressement qui s' opère .

---

man+auto : en ce qui concerne les royaume-uni , m. PRENOM NOM a souligné les difficultés opérationnelles de la zone . cette situation était à l' époque sur du moins 4 en tendance pro forma en pro forma . vous avez vu la publication du chiffre d' affaires du q 1 que nous sommes déjà plus qu' à moins de deux signes du redressement s' opèrent . l' europe du sud , essentiellement l' espagne et le portugal , est très dynamique avec une croissance organique à plus de 5 % .

---

selfsup : les difficultés opérationnelles de la zone ont atteint -1,4 % , consolidées en tendance pro forma , reflétant bien les difficultés opérationnels . vous avez pu constater dans la publication du chiffre d' affaires du q 1 que nous sommes déjà plus qu' à deux signes du redressement qui s' opèrent en europe du sud ; l' espagne et le portugal , très dynamiques avec une croissance organique à plus de 5 % , malgré un ralentissement d' activité hôtelière en espagne .

---

backsum : le royaume-uni a souligné les difficultés opérationnelles de la zone . a l' époque , le chiffre d' affaires du q 1 s' élevait déjà à moins de deux signes du redressement qui règne dans l' europe du sud , essentiellement l' espagne et le portugal , très dynamique , avec une croissance organique à plus de cinq % . malgré PRENOM l' a souligné , le ralentissement de l & d est lié à l' activité en espagne .

---

both : en ce qui concerne le royaume-uni , les difficultés opérationnelles de la zone s' élevaient à -4 % , en tendance pro forma , reflétant bien la difficulté opérationnelle . vous avez vu avec la publication du chiffre d' affaires du q 1 que nous sommes déjà plus qu' à moins de deux signes du redressement qui s' opère . l' europe du sud , essentiellement l' espagne et le portugal , est très dynamique , avec une croissance organique à plus de 5 % , malgré PRENOM l' a souligné , un ralentissement de l ' activité hôtelière en espagne .

---

pred20 : s' agissant du royaume-uni , les difficultés opérationnelles de la zone s' élevaient à -4 % en tendance pro forma , reflétant bien les difficultés opérationnelles . vous avez pu constater , grâce à la publication du chiffre d' affaires du q 1 , que nous sommes déjà plus qu' à moins de deux signes du redressement en europe du sud (essentiellement l' espagne et le portugal) , avec une croissance organique de 5 % (+5 % malgré PRENOM l' a souligné) .

---

inf20 : s' agissant du royaume-uni , les difficultés opérationnelles de la zone ont été présentées à l' époque sur du moins-disant , ce qui constitue bien les difficultés financières . vous avez pu constater que le chiffre d' affaires du q 1 s' établit déjà à moins de deux signes de redressement , dont l' europe du sud , l' espagne et le portugal .

---

predboth : PRENOM NOM a souligné les difficultés opérationnelles de la zone . le chiffre d' affaires du q 1 s' établit à -4 % , en tendance pro forma , reflétant bien la difficulté opérationnelle . le groupe enregistre déjà plus de deux signes de redressement , dont l' europe du sud , l' espagne et le portugal , très dynamique , avec une croissance organique de 5 % . malgré PRENOM a souligné un ralentissement de l' activité hôtelière sur place , notamment sur l' espagne et le portugal .

FIGURE B.3 – Extraits de comptes rendus automatiques n°3

src : PRENOM , nous vous écoutons , monsieur le président , mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et prestataires coupeurs . j' ai le plaisir de vous rendre compte de l' exécution de notre mission au titre de l' exercice deux mille dix-sept . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions , nos rapports au titre de l' assemblée générale ordinaire concernant les comptes et les conventions réglementées , et ce au titre de l' assemblée générale extraordinaire , sont sont demandés dans l' éventualité d' émission d' actions ou d' instruments de capitaux . je vous propose donc de vous présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

tgt : merci , monsieur le président . mesdames , messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et pricewaterhousecoopers , j' ai plaisir de vous rendre compte de l' exécution de notre mission au titre de l' exercice 2017 . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions : le rapport au titre de l' assemblée générale ordinaire concerne les comptes et les conventions réglementées ; ceux au titre de l' assemblée générale extraordinaire sont demandés dans l' éventualité des missions d' action ou d' instrument de capitaux . je vous propose donc de vous présenter ces différents rapports , ainsi que nos conclusions .

---

man+auto : monsieur le président , mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et prestataires et oaktree , j' ai le plaisir de vous rendre compte de l' exécution de notre mission au titre de notre exercice 2017 . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions , nos rapports au titre de l' assemblée générale ordinaire concernant les comptes et les conventions réglementées et ce titre au titre des assemblées extraordinaires , sont demandés dans l' éventualité d' émission d' actions ou d' instruments de capitaux .

---

selfsup : merci monsieur le président . mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et le contrôleur de la dette hlm . je vais vous rendre compte de l' exécution de notre mission au titre d' exercice 2017 . nous avons émis différents rapports , afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions , nos rapports au titre de la assemblée générale ordinaire concernant les comptes et les conventions réglementées . ces rapports sont demandés dans l' éventualité d' émissions d' actions ou d' instruments de capitaux . je vous propose de vous présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

backsum : monsieur le président , mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , des cabinets mazars et des prestataires , j' ai le plaisir de vous rendre compte de l' exécution de notre mission au titre de 2017 . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions . nos rapports au titre de l' assemblée générale ordinaire concernant les comptes et les conventions réglementées au titre de l' exercice 2016 sont demandés dans l' éventualité d' émission d' actions ou d' instruments de capitaux . je vous propose de vous présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

both : merci , monsieur le président . mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et mazars , j' ai le plaisir de vous rendre compte de l' exécution de notre mission au titre d' exercice 2017 . nous avons émis différents rapports , afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions . nos rapports au titre de la assemblée générale ordinaire concernant les comptes et les conventions réglementées , et ce au titre de l' assemblée générale extraordinaire , sont demandés dans l' éventualité d' émissions d' actions ou d' instruments de capitaux . je vous propose de vous présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

pred20 : monsieur le président , mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et les prestataires de rte sont étonnés . j' ai le plaisir de vous rendre compte de la exécution de notre mission au titre de l' exercice 2017 . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions . ces rapports émettront aussi un avis sur les comptes et les conventions réglementées . enfin , les conventions courantes qui ont été présentées lors de la réunion précédente sont liées à l' éventualité d' émissions ou d' instruments de capitaux . je vous propose de présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

inf20 : monsieur le président , mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et le prestataire pricewaterhousecoopers audit , je vous présente l' exécution de notre mission au titre de l' exercice 2017 . nous avons émis différents rapports , afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions . nos rapports émettront également suite à l' assemblée générale ordinaire concernant les comptes et les conventions réglementées . enfin , les conventions courantes qui vous ont été présentées lors de la séance précédente sont posées dans l' éventualité d' émissions ou d' instruments de capitaux . je vous propose de présenter les points essentiels de ces rapports ainsi que nos conclusions .

---

predboth : mesdames et messieurs les actionnaires , au nom du collège des commissaires aux comptes , les cabinets mazars et prestataires concordent avec le résultat de notre mission au titre de l' exercice 2017 . nous avons émis différents rapports afin de vous permettre d' exercer votre jugement au cours du vote sur les résolutions , nos rapports au titre des assemblées générales ordinaires concernant les comptes et les conventions réglementées . ces rapports se sont demandés dans l' éventualité d' émissions ou d' instruments de capitaux . je vous propose donc de présenter les points essentiels de ce rapport ainsi que nos conclusions .

FIGURE B.4 – Extraits de comptes rendus automatiques n°4

src : sur le relais que peuvent apporter les collectivités locales , là aujourd ' hui , cela prend quelle forme ? c' est-à-dire que c' est dans quel type ? c' est dans des forums qu' elles organisent qu' elles peuvent évoquer les challenges , c' est dans leur bulletin , c' est comment ?

---

tgt : quelle forme le relais que peuvent apporter les collectivités locales prend -il ? par quel biais peuvent -elles évoquer le challenge ? par des forums qu' elles organisent , par des bulletins ?

---

man+auto : quelle forme le relais que peuvent apporter les collectivités locales prend -il ? par quel biais peuvent -elles évoquer le challenge ? par des forums qu' elles organisent , par des bulletins ?

---

selfsup : quelle forme peuvent apporter les collectivités locales ? par quel biais peuvent -elles évoquer les challenge ? par des forums qu' elles organisent , par leur bulletin ?

---

backsum : sur le relais que peuvent apporter les collectivités locales , cela prend quelle forme . dans quel type de forums organisent -elles les challenges ?

---

both : le relais que peuvent apporter les collectivités locales prend -il une forme ?

---

pred20 : quelle forme prend le relais que peuvent apporter les collectivités locales ? par quel biais peuvent -elles évoquer les challenges ? par leur bulletin ?

---

inf20 : sur le relais que peuvent apporter les collectivités locales , quelle forme prend -t-il ? par quel biais peuvent -elles évoquer les challenges ? par leur bulletin : comment les bulletins ?

---

predboth : sur le relais que peuvent apporter les collectivités locales , là , cela prend quelle forme . c' est dans des forums qu' elles organisent les challenges .

FIGURE B.5 – Extraits de comptes rendus automatiques n°5

src : toutes les formes . il n' y a pas vraiment de format . cela peut être effectivement par le biais de newsletters , par le biais de bulletins . cela peut être ça peut être simplement pendant des réunions d' information . cela peut être pendant des forums . il n' y a pas de format . on s' adapte . chaque fois qu' on a une demande , on crée des supports qui travaillent pour eux , on leur met à disposition les supports dans le format qu' ils souhaitent pour qu' ils puissent en faire le relais des challenges de manière simple , rapide et efficace et nous les sollicitons en leur demandant de travailler le moins possible et en leur apportant directement les supports . si vous en avez besoin . vous nous dites et nous , on les rédige et on vous les donne .

---

tgt : elles peuvent faire ce relais par plusieurs formes : par le biais de bulletins , pendant des réunions d' information , pendant des forums , etc. . il n' existe pas de format prédéfini . nous nous adaptons . lorsque nous avons une demande , nous leur mettons à disposition des supports dans le format qu' ils souhaitent pour pouvoir faire le relais des challenges de manière rapide et efficace . nous nous efforçons de leur demander le moins de travail possible en leur apportant les supports adéquats . si vous avez besoin , nous pouvons rédiger les supports et vous les fournir .

---

man+auto : ce type de format peut faire par plusieurs newsletters , par le biais de bulletins , pendant des réunions d' information . pendant des forums , il n' existe pas de format prédéfini . nous nous adaptons . lorsque nous avons une demande , nous leur mettons à disposition des supports dans le format qu' ils souhaitent pour pouvoir faire le relais des challenges de manière rapide et efficace . nous les sollicitons , en leur demandant le moins possible et leur apport directement les supports adéquats . si vous avez besoin , nous pouvons rédiger les supports et vous les fournir .

---

selfsup : toutes les formes n' existent pas de format . ils peuvent être par le biais de bulletins , pendant des réunions d' information , durant des forums , etc. . il n' existe pas de format prédéfini . nous nous adaptons . lorsque nous avons une demande , nous leur mettons à disposition des supports dans le format qu' ils souhaitent pour pouvoir faire le relais des challenges de manière rapide et efficace . nous nous efforçons donc de leur demander de travailler le moins possible en leur apportant les supports adéquats . si vous avez besoin , nous pouvons rédiger les supports et vous les fournir .

---

backsum : il n' y a pas vraiment de format . cela peut être par le biais de newsletters , pendant des réunions d' information . nous nous adaptons . a chaque fois que nous avons une demande , nous créons des supports qui travaillent pour eux . nous leur mettons à disposition les supports dans le format qu' ils souhaitent , afin de faire le relais des challenges de manière simple , rapide et efficace . nous les sollicitons en leur demandant de travailler le moins possible et en leur apportant directement les supports . si vous en avez besoin , nous les rédigeons et nous les rédigeons .

---

both : il n' existe pas vraiment de format prédéfini . nous nous adaptons chaque fois que nous avons une demande . nous créons des supports qui travaillent pour eux pour qu' ils puissent faire le relais des challenges de manière simple , rapide et efficace . nous les sollicitons en leur demandant de travailler le moins possible et en leur apportant directement les supports .

---

pred20 : ils peuvent s' organiser par plusieurs formes : par le biais de bulletins , pendant des réunions d' information , durant des forums , etc. . il n' existe pas de format prédéfini . nous nous adaptons . lorsque nous avons une demande , nous leur mettons à disposition des supports dans le format qu' ils souhaitent pour pouvoir faire le relais des challenges de manière rapide et efficace . nous les sollicitons en leur demandant de travailler le moins possible en leur apportant les supports adéquats .

---

inf20 : les outils peuvent être mis à disposition des salariés . ils peuvent également être modifiés par plusieurs formes : forums , forums ... il n' existe pas de format prédéfini . nous nous adaptons . lorsque nous recevons une demande , nous leur mettons à disposition les supports adéquats afin qu' ils puissent prendre le relais des challenges de manière rapide et efficace . nous nous efforçons donc de leur demander de travailler le moins possible en leur apportant les supports adéquats . si vous avez besoin de temps pour répondre à une demande , nous pouvons rédiger les supports et vous les fournir .

---

predboth : il n' existe pas de format prédéfini . nous nous adaptons . chaque fois que nous formulons une demande , nous créons des supports qui travaillent pour eux , afin qu' ils puissent faire le relais des challenges de manière simple , rapide et efficace . nous les sollicitons en leur demandant de travailler le moins possible , en leur apportant directement les supports . si vous en avez besoin , nous rédigeons et nous vous les donnons .

FIGURE B.6 – Extraits de comptes rendus automatiques n°6