MACHINE DE TRADUCTION AU NIVEAU DES DOCUMENTS

Rapport de stage

29 août 2022

Michael Junior FOTSO FOTSO

Tuteur: Alexis MATHEY — Pirashnth RATNAMOGAN
Supervisor: Oana Balalau

ÉCOLE POLYTECHNIQUE

INSTITUT POLYTECHNIQUE DE PARIS

BNP PARIBAS

Remerciements

Je ne saurais rédiger ce rapport sans remercier mon tuteur de stage Alexis Mathey pour sa réactivité et ses conseils précieux face à toute situation. Son encadrement sans faille m'a permis d'en apprendre le maximum tout au long de mon stage.
Je tiens également à remercier Pirashanth, que j'ai toujours considéré comme un second tuteur de stage. Son implication dans mes missions tout au long de mon stage m'a permis de toujours rester dans la bonne direction.
Je tiens également à remercier tous ceux qui m'ont encadrés, en particulier William, Tom, Joel, Mhamed, Laurent ainsi que tous les autres de l'équipe pour les conseils qu'ils m'ont prodigués.
Je remercie également les autres stagiaires, qui ont parfois su jouer des rôles d'aîné à mon égard et qui m'ont légué plusieurs techniques en lien ou non avec mon stage.

Contexte

Malgré les performances sans cesse croissantes des machines de traduction de nos jours, la traduction des documents dans leur globalité demeure encore un enjeu de taille. En effet, la plupart des méthodes de traductions actuelles effectue la traduction phrase après phrase. Cela les empêchent de prendre en compte le contexte global d'un document au cours de sa traduction. Il en ressort alors très souvent des erreurs de cohérence lexicale et sémantique, des pronoms personnels et des noms propres mal traduits, même avec les meilleures méthodes existantes. Partant de ce constat, le but de mon stage était d'explorer et d'implémenter quelques approches de traduction au niveau des documents ainsi que de faire une analyse poussée des résultats obtenus. De plus, mes modèles devaient idéalement pouvoir s'intégrer à l'environnement informatique existant au sein de l'entreprise. Les deux principales approches qui ont fait l'objet de mon étude était l'utilisation des « Hierarchical attention network » et du « Multi-Resolutional Learning ».

Sommaire

Table des figures

Liste des sigles

HAN : Hierarchical Attention Network

MR : Multi-Resolutional learning

NMT : Neural Machine Translation

RNN : Recurrent Neural Network

CNN : Convolutional Neural Network

BPE : Byte Pair Encoding

NLP : Natural Language Processing

Introduction

Le Data Lab de l'Analytic consulting group de la BNP est une entité qui développe des solutions d'IA utilisées par d'autres parties de l'entreprise. On peut citer entre autres des solutions d'extraction de mots-clés dans des documents (Doc Factory) pour accélérer le traitement des factures, des solutions de reconnaissance vocale pour faciliter les détections de fraude, ainsi qu'une machine de traduction pour permettre à toute l'entreprise de faire des traductions tout en garantissant la confidentialité des données. Ces solutions sont développées en interne pour garantir une sécurité des données et éviter qu'elles ne soient divulguées à des entités externes. C'est pourquoi il est important pour eux de développer des produits qui rivalisent avec les meilleures solutions externes. D'où la recherche active dans cette équipe.
En ce qui concerne la machine de traduction, elle a souvent essuyé certaines critiques de la part des utilisateurs. En effet, les principaux textes traduits étant des documents, on s'apercevait assez vite des limites de leur modèle qui avait du mal à inclure le contexte global d'un document au cours de sa traduction. Cela se traduisait par des problèmes de cohésion lexicale, d'ellipse, etc... Il était donc important pour eux de développer des solutions capables de surmonter ce genre de problèmes. Bien que le système qu'il utilise fasse partie des modèles de l'état de l'art, il n'en demeure pas moins qu'elle se contente de traduire phrase après phrase, ce qui la rend intrinsèquement incapable d'avoir une vue globale sur le document.
L'objectif de notre stage était donc d'implémenter des solutions permettant la traduction au niveau des documents. Notre point de départ était le « Hierachical attention network » (HAN) proposé par [1] . Après l'avoir implémenté et testé sous plusieurs facettes, certaines de ses limites ont été mise en évidence. C'est à partir de ces failles, qu'est née l'idée de tester l'approche du « Multi-Resolutional Learning » (MR) décrit par [2].
Dans ce rapport, nous allons commencer par présenter des concepts fondamentaux du « Natural Language Processing » (NLP), en particulier ceux liés aux machines de traduction et des méthodes que nous avons utilisées. Nous allons ensuite présenter un aperçu de l'évolution ainsi que l'état de l'art des machines de traduction tant au niveau des phrases que des documents. Nous allons enchaîner avec une description plus précise des méthodes que nous avons utilisés combinées aux résultats pertinents qui ont été obtenus. Enfin, nous présenterons quelques champs d'amélioration que nous n'avons pas pu investiguer profondément.

1. Concepts fondamentaux utilisés

Tout au long de notre stage, nous avons implémenté plusieurs modèles de machine de traduction. Il s'agit en l'occurrence des HAN et du MR. Leur compréhension nécessite la maîtrise de plusieurs autres concepts du NLP, notamment les « Transformer », l'« Embedding » et les réseaux hiérarchiques. Pour les utiliser, il faut avoir un dataset bien construit et prétraité à travers une série d'étapes. Enfin, pour évaluer les modèles, il faut des métriques pertinentes et adaptées au contexte. Avant d'éclaircir chacune de ces notions, il est important de définir scientifiquement ce qu'est une machine de traduction.

1.1 Machine de traduction

Les machines de traduction sont des systèmes qui permettent de traduire des textes d'une langue source vers une langue cible. La plupart d'entre elles fonctionne phrase après phrase tandis que d'autres essayent tant bien que mal d'introduire des informations contextuelles.

1.1.1 Les machines de traduction phrase par phrase

Si on appelle $X = (x_1, x_2, .., x_n)$ une phrase dans la langue source, $Y = (y_1, y_2, \dots, y_m)$ la phrase correspondante dans la langue cible et $Y' = (y'_1, y'_2, \dots, y'_T)$ la phrase traduite, une machine de traduction essaye de maximiser la probabilité $p(Y'|X)$ à l'inférence. Cependant, étant donnée qu'au cours de la traduction les mots y'_i sont générés les uns après les autres, on ne peut qu'avoir une estimation de $p(Y'|X)$ qu'on essaye de maximiser. La méthode gloutonne consiste à maximiser $\prod_{i=1}^{T} p(y'_i|y'_{j<i}, X)$. D'autres techniques comme Beam search [3] est utilisé.

1.1.2 Les machines de traduction au niveau des documents

En gardant les mêmes notations que dans la partie précédente, appelons C_X un contexte dont dépend la phrase X, C_Y un contexte dont dépend la phrase Y et C'_Y pour Y'. Les contextes C_X et C_Y peuvent être des phrases précédentes et/ou suivantes de X ou Y. L'idée cette fois-ci est de maximiser l'une des probabilités suivantes à l'inférence :

- $p(Y'|X, C_X)$: systèmes basés sur le contexte de l'encodeur
- $p(Y'|C'_Y, X)$: systèmes basés sur le contexte du décodeur
- $p(Y', C'_Y|X, C_X)$: systèmes basés sur le contexte de l'encodeur et du décodeur
- $P(C'_Y|C_X)$: systèmes qui génèrent le document globalement

Le HAN [4] se retrouve dans les trois premiers cas tandis que le MR dans le dernier.

1.2 Le « preprocessing »

Le preprocessing des données est une étape fondamentale dans toute tâche du NLP. En effet, selon la méthode choisie, les écarts de performances peuvent être impressionnants. Le preprocessing désigne toutes les transformations appliquées au texte brut avant que celui-ci ne soit envoyé au modèle.

1.2.1 La normalisation des ponctuations

Réalisée à l'aide du script de Moses [4], elle permet de normaliser toutes les ponctuations dans un ensemble prédéfini et propre à chaque langue. Cela permet de remplacer les ponctuations inconnues par des caractères spéciaux. C'est en général la première étape du preprocessing.

1.2.2 La « tokenization »

Toujours réalisée à l'aide du script de Moses [4], elle permet de séparer le texte en ses mots. Elle permet en outre de décoller les ponctuations à la fin de certains mots. Cela permet donc d'avoir un vocabulaire plus petit et plus cohérent.

1.2.3 Le « truecaser »

Faisant également partie des scripts de Moses [4], il permet de faire commencer toutes les phrases par des minuscules tout en conservant les majuscules sur les noms propres. C'est pourquoi il a besoin d'avoir une représentation statistique des mots du texte afin de discerner entre les noms propres et les noms communs. Il est important, car il réduit assez la taille du vocabulaire finale.

1.2.4 Le nettoyage du corpus

Quand on fait la traduction, il est important que le ratio des longueurs entre la phrase source et sa traduction soit proche. Le but de cette étape est donc de supprimer les phrases trop longues par rapport à leur équivalent dans la langue source ou la langue cible. Elle permet aussi de supprimer les lignes vides.

1.2.5 Le « Byte Pair Encoding » (BPE)

Le BPE [5] est une méthode de tokenization au niveau des sous-mots. En effet, elle décompose chaque mot du texte en sous parties appartenant à son vocabulaire. Un caractère spécial permet de reconnaître les mots coupés des espaces préexistants entre les mots.
Une étape importante dans l'usage du BPE est la constitution de son vocabulaire de sous-mots. Elle commence par un vocabulaire constitué de tous les caractères

disponibles, elle agrège progressivement les éléments de son vocabulaire deux à deux par ordre de fréquence d'apparition jusqu'à ce que la taille de son vocabulaire soit la taille définie. Considérons par exemple la chaîne $s = aaabdaaabac$. La paire aa est la plus fréquente donc elle sera encodée par un nouveau caractère $Z = aa$ et aura $s = ZabdZabac$. La nouvelle paire la plus fréquente est $Y = ab$ et aura $s = ZYdZYac$. Ensuite, la prochaine paire est $X = ZY$ et on aura $s = XdXac$. Comme il n'y a plus de paire qui apparaît plus d'une fois, la compression s'arrête là. La décompression se fait par remplacement dans l'ordre inverse.
L'avantage de cette méthode est qu'elle rend le modèle robuste aux nouveaux mots. Dans l'extrême, un mot peut toujours être décomposé en caractères.

1.3 Les modèles

Dans cette partie, nous allons présenter les modèles avec lesquels nous avons interagis, de près ou de loin.

1.3.1 L'« Embedding »

Il s'agit d'une étape fondamentale dans toute tâche de NLP. En effet, étant donné que les ordinateurs ne savent que manipuler des chiffres, il est indispensable de transformer chaque mot du texte en un vecteur de $R^{d_{emb}}$ avec d_{emb} la dimension de l'embedding. Il existe plusieurs façons de faire cette correspondance :

- *One hot encoding :* chaque mot est représenté par un long vecteur de R^V avec V la taille du vocabulaire. Le vecteur comporte 1 à l'indice du mot et 0 partout ailleurs. Cette façon de faire est très médiocre car on ne dispose d'aucune information sémantique par rapport aux mots.
- *Embedding pré-entrainés :* Il s'agit d'importer un gros dictionnaire qui fait la correspondance entre chaque mot (de son vocabulaire) et un vecteur. Ces vecteurs ont été obtenus à l'aide d'un pré-entraînement non supervisé. Les méthodes de pré-entraînement populaires sont *skipgram* [6] qui essaye de prédire le contexte à partir d'un mot et *cbow* [6] qui lui à l'inverse essaye de prédire le mot à partir du contexte. Ces méthodes sont regroupées dans le Framework *Word2vec* [6]. Nous avons également des embedding de très bonne qualité comme ceux fournis par des modèles tel que *BERT* [7]. Le pré-entraînement utilisé est le « masquage language model », qui consiste à prédire des mots masqués aléatoirement dans le texte. Toutes ces méthodes ont le mérite de comprendre des relations sémantiques entre les mots.
- *Embedding supervisé :* Il s'agit des embedding qui sont construits pour s'adapter à un problème précis dont on dispose à priori d'un modèle (classification, traduction, etc.) Très souvent, une matrice de paramètres de dimension $R^{V \times \mathrm{d_{emb}}}$ est introduite à l'entrée du modèle. Il est important de remarquer que chaque ligne de la matrice correspond à un mot. Les poids de la matrice sont initialisés aléatoirement et sont réajustés au fur et à mesure de l'entraînement avec la

rétro-propagation des gradients. Cette méthode est la plus utilisée quand le dataset est conséquent ou quand le problème est trop spécifique. C'est cette façon de procéder que nous avons utilisé dans tous nos modèles.

1.3.2 Transformer

Transformer est un modèle qui a vraiment révolutionné le monde de la traduction automatique. Décrit pour la première fois par [8], il demeure aujourd'hui l'un des modèles les plus implémentés en traduction automatique. C'est un excellent compromis entre complexité et efficacité. En outre, c'est ce modèle que nous avons utilisé comme « Baseline » dans toutes nos expériences.
Comme décrit par [8], transformer est le tout premier modèle « seq-to-seq » [1] qui repose entièrement sur le mécanisme d'attention. Cela lui permet de traiter tous les mots d'une séquence en même temps, contrairement à la plupart des modèles « seq-to-seq ». Il peut également paralléliser le traitement des mots d'une séquence, ce qui lui donne un réel avantage en termes de temps d'entraînement et d'inférence quand les GPU[2] sont disponibles.
Si on lui passe en entrée une séquence d'« embedding » de mots $X = (x_1, x_2, \dots, x_n)$, avec $x_i \in R^{d_{model}}$, le modèle va appliquer une « self-attention » entre la « query » $q = x_i$, et les « keys » $K = (x_1, x_2, \dots, x_n)$ dont les « values » sont $V = (x_1, x_2, \dots, x_n)$ grâce à la formule :

$$\text{attention}(q, K, V) = softmax(\frac{qK^T}{\sqrt{d_{model}}})V$$

Pour paralléliser le traitement de chaque mot, la query devient $Q = (x_1, x_2, \dots, x_n)$ et on a :

$$\text{attention}(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V$$

En général, l'attention est remplacée par une « multihead-attention », qui consiste à diviser la dimension du modèle en sous dimension pour augmenter le champ de perception du modèle. Si k est le nombre de « head », la dimension des sous espaces est $d_k = \frac{d_{model}}{k}$. Ainsi on a :

$$multihead\text{-}attention\,(X) = Concat(head_1, .., head_k)W_O$$
$$\text{avec } head_i = attention(QW_i^Q, K_i^K, V_i^V)$$

$W_i^Q, K_i^K, V_i^V \in R^{d_{model} \times d_k}$ et $W_O \in R^{d_{model} \times d_{model}}$ sont des matrices de paramètres.

[1] Modèle qui prend une séquence en entrée et qui retourne une séquence en sortie
[2] Processeur graphique utilisé très utilisé en apprentissage profond

Par rapport à l'architecture, elle est composée d'un encodeur et d'un décodeur comme la plupart des modèles seq-to-seq. L'architecture peut être résumée par la Figure 1.

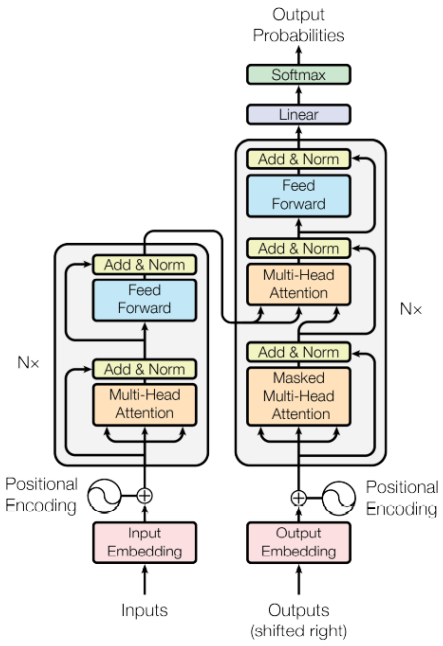

Figure 1: Architecture Transformer

1.3.3 Le HAN

Proposé par [1], ce modèle constituait le véritable point de départ de mon stage. En effet, c'est un modèle de traduction au niveau des documents qui traduit chaque phrase en essayant d'inclure le contexte des k phrases précédentes. Avec k un paramètre du modèle. La première mission de mon stage était donc de produire une implémentation simple de ce modèle de sorte à ce qu'il puisse facilement s'intégrer aux systèmes en production à la BNP.
Le HAN peut être vu comme un modèle qui se greffe sur l'encodeur et/ou le décodeur d'un modèle de traduction déjà existant. C'est pourquoi notre implémentation devait être compatible avec les modèles de traduction déjà en production à la BNP.
Ainsi, appelons h_j^i la sortie de l'encodeur et/ou décodeur d'une machine de traduction standard pour le $i^{ième}$ mot de la $j^{ième}$ phrase et $h_j = (h_j^1, .., h_j^n)$ la phrase j. Lors de la traduction de la phrase j, le HAN suit les étapes suivantes :

- *L'attention au niveau des mots :* pour chaque mot h_j^i de la phrase j et pour chacune des k phrases précédentes $h_{j-1}, h_{j-2}, \dots, h_{j-k}$, une attention est effectuée entre le mot h_j^i et les mots de la phrase h_{j-p} avec $p \in [1 \dots k]$. cela permet d'avoir k représentations du mot h_j^i où chacune d'elle montre l'importance de la phrase $j-p$ par rapport au mot i de la phrase j. Elle peut être résumée par la formule suivante :

$$q_j = f_w(h_j)$$

s_{j-p}^{j} =MultiHead-attention($Q = q_j, K = h_{j-p}, V = h_{j-p}$) pour $p \in [1 \dots k]$

Avec f_w une transformation linéaire pour obtenir la « query », $s_{j-p}^{j} = (s_{j-p}^{j,1}, s_{j-p}^{j,2}, \dots, s_{j-p}^{j,n})$ est la nouvelle représentation de la phrase j, qui tient compte de l'importance de phrase $j - p$.

- *L'attention au niveau des phrases :* pour chaque mot h_j^i, la représentation finale est obtenu en faisant une attention entre le mot h_j^i et ses k représentations $s_{j-1}^{j,i}, s_{j-2}^{j,i}, \dots, s_{j-k}^{j,i}$. Cela permet pour chaque mot de savoir sur lesquelles des phrases précédentes il faudrait plus ou moins prêter attention. Elle se résume par la formule précédente :

$$q_j = f_s(h_j)$$
$$d_j^i = \text{Multihead-attention}(Q = q_j^i, K_j^i, V_j^i) \text{ pour } i \in [1, \dots, n]$$

Avec $K_j^i = V_j^i = (s_{j-1}^{j,i}, s_{j-2}^{j,i}, \dots, s_{j-k}^{j,i})$ et f_s une autre transformation linéaire pour obtenir la query. A l'implémention, il existe des façons de vectoriser le calcul afin de traiter tous les mots en parallèle. d_j^i est la représentation finale de l'entrée h_j^i qui tient compte des k phrases précédentes.

- *Le « context gating » :* Le but est de créer un portail afin de sélectionner la quantité d'information qui vient du contexte d_j par rapport à l'entrée initiale h_j. Cette étape se résume par la formule :

$$\lambda_j = sigmoid(W_h h_j + W_d d_j)$$
$$\tilde{h_j} = \lambda_j h_j + (1 - \lambda_j) d_j$$

Avec W_h et W_d des matrices de paramètres et $\tilde{h_j}$ la représentation finale de h_j. Le figure 2 résume les étapes.

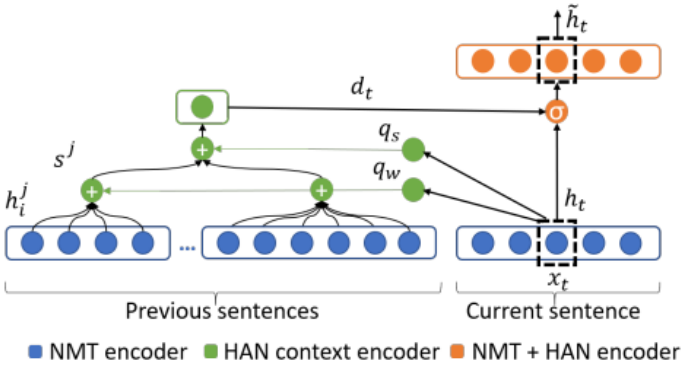

Figure 2: Architecture HAN

1.3.4 Le MR

Le MR est plus une méthode d'apprentissage qu'un modèle. En effet, proposée par [2], les auteurs partent du constat que les méthodes qui se basent sur le contexte ne sont pas toujours efficaces. En effet, comme nous le verrons plus loin dans ce rapport, certaines des améliorations de ces types de modèle ne sont pas toujours interprétables par des informations contextuelles.
Une autre solution serait donc de traduire un document dans sa globalité. C'est-à-dire percevoir les documents comme des très longues phrases et d'essayer d'entraîner des modèles phrase par phrase performants dessus. Cependant, des études telles que [9] [10] ont montré que cette méthode conduisait à des résultats catastrophiques.
C'est à partir de ces observations que la méthode MR à vue le jour. Elle consiste à créer un dataset en scindant un document en sous-blocs. Si le document possède k phrases, elle va créer k blocs de 1 phrase, E(k/2) blocs de 2 phrases, E(k/3) bloc de 3 phrases,.., 1 bloc de k phrases. Un bloc peut être vu comme la concaténation des phrases qui la constituent. Cela permet donc au modèle d'être aussi bon sur les phrases courtes que sur les longues phrases.
Cependant, il est important de noter que cette méthode augmente la taille du dataset et donc le temps et le coût de calcul. En effet, si on appelle d la taille moyenne d'un document et n le nombre de mots du document, la taille finale est de :

$$\tilde{n} \cong n * d * \frac{\pi^2}{6}$$

Ainsi, étant donné ce problème, on ne pouvait se permettre de travailler avec des documents de taille arbitrairement grande. C'est pour cette raison qu'il était important de définir une taille maximale par document et de scinder les documents excédant cette taille. Dans la suite, on va appeler MR(d) un modèle MR entraîné avec des documents de taille maximale d.

1.4 Les métriques d'évaluation

Une fois qu'on a des modèles fonctionnels, il est important de les évaluer. Il existe beaucoup de métriques dans le domaine de la traduction automatique dont l'une des plus populaires est le BLEU [11]. Bien que très corrélée au jugement humain (plus de 0.95 de coefficient de corrélation linéaire), elle reste toutefois insuffisante pour déceler les particularités de la traduction au niveau des documents. En effet, les problèmes de cohérence lors de la traduction des pronoms personnels ou des noms propres ne seront pas détectés.
Considérons par exemple le paragraphe suivant :
fr:
Marilyne aime beaucoup la musique.
C'est pourquoi **ses** écouteurs sont en bon état.
Traduction en:

Marilyne likes music a lot.
That is why **their** headphones are in good condition
Reference en:
Marilyne likes music a lot
That is why **her** headphones are in good condition

Dans le cas présent, la traduction aura un bon score de BLEU. Cependant, il est clair que la traduction ne tient pas compte du sujet « Marilyne » de la phrase précédente (*their* au lieu de *her*). C'est pourquoi une métrique spécifique aux documents doit pouvoir sanctionner particulièrement ce genre de phénomène.

1.4.1 Le BLEU score et ses variantes

Le BLEU score est une métrique introduite par [11]. Intuitivement, elle attribue un score en fonction du nombre de mots et/ou groupe de mots de la traduction qui correspond à la référence. Ses valeurs varient de 0 si aucun mot ne correspond à 100 si la traduction est **exactement** la référence.
Plus spécifiquement, elle peut être résumée par la formule suivante pour une phrase:

$$BLEU = BP.\exp(\sum_{n=1}^{N} w_n \log p_n)$$

Avec :

- BP : un terme qui pénalise les phrases candidates trop courtes par rapport à la référence. En effet, si la phrase candidate à peu de mots, elle a plus de chance de correspondre à la référence qu'une phrase avec plus de mots.
- N: C'est le nombre de maximal de n-gram[3]. En effet, les correspondances entre mots seront évaluées pour les n-gram pour n $\in [1 \dots N]$.
- p_n : Le score obtenu pour la correspondance des n-gram. Pour $n = 1$, c'est le ratio du nombre de mots de la phrase candidate qui correspond à la référence. Cependant, il y'a une petite subtilité. Considérons les deux phrases suivantes :
  Candidate: bonjour bonjour bonjour bonjour
  Référence : bonjour comment tu vas ?
  Les deux phrases ont le même nombre de mots donc on aura $BP = 1$. De plus, le nombre de mots du candidat qui correspond est 4 (soit la totalité) et le nombre total de mots dans le candidat est 4. Donc le ratio p_1 vaut 1. Pourtant, il est évident que la traduction est médiocre. C'est pourquoi il existe une astuce « *modified n-gram précision* » qui consiste à bloquer le nombre maximal de correspondances au nombre maximal d'apparitions du mot dans la référence. Dans le cas présent, on va passer de 4 à 1 et le score va donc varier de 1 à $\frac{1}{4}$.
- w_n : C'est le poids que l'on accorde au n-gram.

[3] Ce sont les groupes de mots. Par exemple 1-gram se sont des mots, 2-gram des doublets...

L'un des problèmes de la métrique BLEU est qu'elle dépend du preprocessing. En effet, en fonction de la gestion de la casse, des ponctuations, etc... les résultats avec la métrique BLEU peuvent fortement varier. C'est ce que révèle [12] lorsqu'il a mis en place la métrique sacrebleu. Elle comporte un preprocessing intégré afin d'unifier les résultats. Par la suite, nous ferons référence à cette métrique par « s-bleu ».
Dans le papier [10], une variante du BLEU appelé d-bleu a été créée pour évaluer la traduction au niveau des documents. Elle consiste à appliquer le BLEU sur la concaténation de toutes les phrases du document. Elle serait plus pertinente pour l'analyse des documents. Cependant, nous avons toujours trouvé cette métrique fortement corrélée au s-bleu. Dans la suite, nous la référencerons par « d-Bleu ».

1.4.2 Le « Accuracy of Pronoun Translation » (APT)

C'est une métrique d'évaluation spécialisée pour les documents proposée par [13]. Elle permet de voir à quel point la traduction est précise au niveau des pronoms. Le choix de cette métrique a été motivé par les travaux de [14] qui montre que 86.73% des problèmes dans la traduction des documents proviennent soit des ellipses (omission d'un ou de plusieurs mots) ou des problèmes d'ambigüité. La traduction des pronoms et des noms propres représente une grande partie des problèmes évoqués précédemment. Son principe de fonctionnement est plutôt simple mais efficace. En effet, partant d'une liste de pronoms prédéfinis de langue source et de langue référence, elle repère tous les pronoms dans le document initial, le document traduit et le document de référence. Ensuite, elle fait un alignement entre tous les mots du document source et du document traduit ainsi qu'entre tous les mots du document source et du document de référence. L'outil qui nous permettait de faire cette correspondance était « fast-alignment » [15]. Une fois les alignements effectués, on dispose d'une série de triplets (pronom source, pronom traduit, pronom référence). Un score entre 0 et 1 est attribué selon que le pronom traduit soit le pronom référence (1), selon que l'un des pronoms est inconnu (hors de la liste prédéfinie), selon que les pronoms traduit et référence sont différents (0), etc... Le score final est la moyenne des scores de chaque triplet multiplié par 100.

1.4.3 D'autres métriques liées aux documents

Il existe plusieurs autres métriques liées aux documents. Cependant, la plupart de ces métriques nécessitaient une ré-implémention intégrale. C'est pourquoi nous n'avons pas pu les exploiter dans nos travaux. On peut citer entre autres la métrique BlonDe proposée par [14], c'est une métrique tout en un qui évalue plusieurs aspects des documents tels que les problèmes d'ellipse, d'ambiguïté, de cohérence lexicale et sémantique.
Nous avons également des métriques plus spécifiques à la cohésion et la cohérence. Par exemple la métrique proposée par [16] qui calcule le ratio entre le nombre de « mots de contenus » répétés sur le nombre total de « mots de contenus » du documents.

1.5 Les datasets et l'itérateur

Pour entraîner un modèle de traduction au niveau des documents, il faut avoir des documents bilingues et en grande quantité. Cependant constituer un tel dataset n'est pas toujours une chose évidente. De plus, la façon d'itérer sur ce dataset n'est pas similaire à un itérateur classique phrase après phrase. En effet, les documents doivent être contigus dans les batchs[4] et le mixage doit se faire à l'échelle des documents.

1.5.1 Les datasets

Il existe plusieurs petits datasets dédiés à la traduction des documents. En effet, ces datasets sont uniquement destinés à des fins expérimentales pour évaluer les performances des modèles. Ils sont de l'ordre 0.1M phrases pour l'ensemble d'entraînement et 1K pour l'ensemble de test et de développement. On peut citer entre autres :
IWSLT TED [17] *:* il s'agit des conférences Ted X traduites en plusieurs langues. C'est un dataset très pertinent pour la traduction au niveau des documents car chaque document est un véritable discours humain. Dans notre stage, nous nous sommes focalisés sur les datasets *IWSLT TED 2014 ES-EN* et *IWSLT TED 2015 EN-FR*
Open subtitle : Il s'agit des sous-titres de films et séries traduits en plusieurs langues. Certes, il y a une cohérence globale, mais le changement constant d'interlocuteur le rend un peu plus challengeant que le précédent dataset.
WMT : C'est un dataset très populaire pour l'évaluation de la traduction en général. Il est composé des données provenant de plusieurs sources. Cependant il existe également des versions dédiées à la traduction de documents.

Lorsqu'il faut passer à l'échelle de la production, obtenir des données devient beaucoup plus compliqué. En effet, trouver un dataset d'une taille de 200M phrases constitué de documents cohérents n'est pas une chose facile. Nous avons donc appliqué la stratégie suivante :

- Repérer les gros datasets cohérents : *Europarl* (texte de parlement), *DGT* (texte de l'union européenne), *ECB* (texte de la banque centrale), *JRC* (texte de lois), *MultiUN* (document de l'Union européenne), etc... tous ces datasets sont de bonne qualité, mais ne possèdent pas une séparation au niveau des documents.
- Ensuite, on définit une taille de document d (par exemple 200 phrases par document) et on suppose une cohérence par bloc de d phrases dans chaque dataset séparément.
- On divise ensuite les datasets par blocs de document de taille d.
- On fusionne les blocs provenant de tous les datasets choisis initialement.
- On mélange tout le dataset à l'échelle des documents de taille d .

[4] C'est un paquet de données qui est traité globalement par le modèle.

Cette stratégie nous a permis d'avoir un dataset de documents pouvant être utilisé pour entraîner un modèle destiné à la production et également de le challenger avec les modèles déjà en production au sein de l'entreprise.
Dans la suite, appelons *data(d)* le dataset obtenu en appliquant la stratégie précédente sur le dataset *data ; data* peut-être un dataset unique ou un ensemble de datasets.

1.5.2 L'itérateur

Comme tout modèle d'apprentissage profond, l'entraînement de nos modèles requiert plusieurs « epochs » (passage entier sur le dataset) chaque epoch étant constitué de plusieurs pas (un « forward » et « backward » sur un batch de données). Le batch à une taille maximale *batch_size* prédefinie. L'itérateur désigne donc le processus de la constitution des batchs afin de parcourir tout le dataset.
Par défaut, à une epoch donnée, l'itérateur mélange toutes les phrases, puis les envoie une à une jusqu'à remplir un batch, et continue ainsi jusqu'à parcourir tout le dataset. Cependant, cette façon de faire est inappropriée lorsqu'il s'agit des documents car dans un batch on aura des phrases provenant de divers documents.
La solution était donc de mélanger à l'échelle des documents et non à l'échelle des phrases à chaque epoch. De plus, il fallait passer l'information sur le document auquel appartient chaque phrase dans le batch. En effet, il pourrait exister des batchs constitués de la fin d'un document et du début d'un autre. L'information du document permet donc de savoir où prêter attention lors des attentions hiérarchiques.

1.6 L'implémentation

L'implémentation de toutes ces méthodes était également un enjeu important de mon stage. En effet, étant donné que mes modèles étaient censés s'intégrer à ce qui se faisait déjà dans l'équipe, il fallait qu'ils soient implémentés avec le même Framework utilisé par la BNP, à savoir : Open Neural Machine translation (OpenNMT) [18]. De plus, l'environnement informatique était également un challenge intéressant.

1.6.1 Le Framework OpenNMT

OpenNMT [18] est un Framework spécialisé pour la conception des machines de traduction. Sa modularité le rend également très efficace pour la recherche.
Nous avons travaillé avec sa dernière version *OpenNMT 2.0*, qui repose entièrement sur la librairie *Pytorch*. Elle nous permettait de paramétrer avec finesse tous les compartiments du modèle (embedding, encodeur, décodeur, ...), de personnaliser les méthodes d'optimisation et de manipuler correctement les données (preprocessing, itérateur, batch).

De plus, il dispose de plusieurs accessoires utiles tels que la visualisation de l'entraînement via tensorboard, la traduction et l'évaluation après entraînement et une bonne gestion de la sauvegarde des modèles au cours de l'entraînement.

1.6.2 L'environnement informatique

Tous nos codes devaient être produits sur une plateforme appelée *Datacamp*. Elle met à notre disposition des machines virtuelles disposant d'un environnement python pré configuré. Les ressources des machines virtuelles étaient allouées dynamiquement selon les disponibilités dans les nœuds de l'environnement distribué. Ainsi, l'accès concurrent aux ressources par tous les utilisateurs ne rendait pas toujours le travail simple car il fallait parfois attendre qu'un GPU se libère pour pouvoir l'utiliser.
Néanmoins, un des avantages de cet environnement est que les données étaient également partagées, ce qui me permettait d'avoir accès facilement aux données ou aux modèles pré entraînés dont j'avais besoin.
Il est également important de mentionner la plateforme *BitBucket* qui est un équivalent de *GitHub,* mais interne à l'entreprise. Elle me permettait donc de gérer les différentes versions de mon code et de le faire valider par mes tuteurs de stage avant la fusion sur le répertoire principal.

2. Aperçu du monde de la traduction automatique

La traduction automatique est toujours un domaine actif de la recherche. Il a connu plusieurs évolutions jusqu'à nos jours et il réserve encore de belles surprises. En ce qui concerne la traduction au niveau des documents, plusieurs études ont été menées, mais le déclic n'a pas encore été atteint. En effet, les résultats d'état de l'art restent toujours entre les mains des modèles de traduction au niveau des phrases.

2.1 La traduction au niveau des phrases

Les premières méthodes de traduction automatique ont vu le jour dans les années 1949 et étaient basées sur la théorie de l'information. Ce n'est qu'à la fin des années 80 que les chercheurs d'IBM on introduit la traduction statistique à travers [19]. Il s'agissait d'une approche où les traductions sont générées sur la base d'un modèle statistique dont les paramètres sont obtenus à partir d'une analyse sur le texte bilingue. Avec la monté en flèche des réseaux de neurones au cours de la dernière décennie, le monde de la traduction automatique à connu un nouveau tournant. On est passé de la traduction statistique au « Neural machine translation » (NMT) qui signifie machine de traduction à réseaux de neurones.
Cependant, les premiers modèles NMT ne se différenciaient réellement que par le fait qu'ils représentaient les mots dans un espace vectoriel continu. Les premiers systèmes modélisant les séquences de mots étaient basés sur les « recurrent neural network » (RNN). Les RNN bidirectionnels étaient en général utilisés pour l'encodeur comme le montre [20]. Ces modèles souffraient de plusieurs problèmes tels que les gradients évanescents et l'oublie des informations lorsque la séquence devenait longue.
Plusieurs solutions ont été apportées pour palier à ces problèmes tel que l'usage du mécanisme d'attention qui permet au décodeur de se concentrer sur des parties précises des entrées lorsqu'il génère les mots [20, 21].
Les « Convolutional Neural Networks » (CNN) ont également été utilisés pour pallier aux problème des RNN. Lorsqu'il a été combiné au mécanisme d'attention comme le montre [22] , ça a été un franc succès.
Avec l'arrivée des Transformer [8], le monde de la traduction a subi un autre tournant majeur. En effet, comme décrit dans la partie précédente, il s'agissait du premier modèle capable de traiter les séquences de façon parallèle. De plus, il était basé entièrement sur le mécanisme d'attention. Il était le modèle état de l'art sur plusieurs datasets dès sa sortie.
Ainsi, grâce à sa simplicité, les chercheurs ont pu développer de nombreux modèles de traduction automatique basés sur les Transformer et peu coûteux en ressource [23].
En ce qui concerne l'état de l'art, les performances des modèles dépendent parfois de la paire de langue choisie, néanmoins, il existe des modèles qu'on retrouve toujours en tête de liste. Le tableau 1 montre le classement actuel des modèles sur le dataset *WMT EN-FR*.

Rank	model	Bleu score	article
1	Transformer +BT	46.4	[24]
2	Noisy back-translation	45.6	[25]
3	mRASP + Fine Tune	44.4	[26]
4	Transformer + R-Drop	43.95	[27]
5	Transformer (ADMIN init)	43.8	[28]

Tableau 1; Top 5 modèle NMT état de l'art

Nous avons choisi de présenter la paire de langue *EN-FR* mais le classement est sensiblement pareil pour les autres paires de langue. De plus, remarquons que tous ces modèles sont basés sur Transformer.
En outre, la figure 3 nous montre l'évolution des publications en NMT. On remarque un changement de rythme clair en 2016, date de sortie des Transformer.

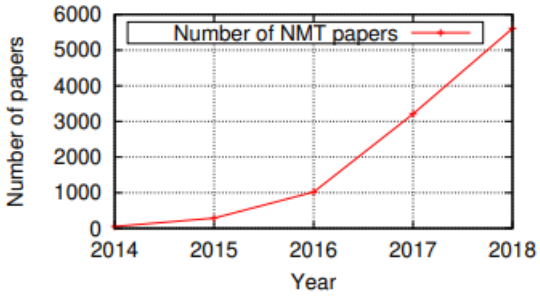

Figure 3: Evolution des publications avec le temps

2.2 La traduction au niveau des documents

Dans le monde de la traduction de documents, plusieurs approches ont été tentées. Certaines essayant d'inclure des informations provenant du contexte de la source, d'autres des informations provenant de la cible et parfois une conjugaison des deux.
Les premiers modèles qui utilisaient les informations de la source ont essayé de l'inclure dans l'encodeur. C'est le cas de [29] qui a modifié l'architecture des RNN pour inclure une couche d'attention supplémentaire afin de tenir compte des phrases précédentes. Cette approche fournissait des améliorations sur des petits dataset mais restait médiocre sur des gros datasets. [30] a proposé le premier modèle NMT dépendant du contexte apportant de réelles améliorations par rapport aux modèles phrase après phrase. Ils ont employé deux niveaux de RNN hiérarchiques pour résumer les informations des phrases précédentes. [31] a modifié la structure par défaut des Transformer pour inclure les informations des phrases précédentes. En effet, les $N-1$ premières couches de transformer prenaient en entrée la phrase précédente et la dernière couche faisait une attention entre la phrase courante et la sortie des $N-1$ premières couches. Le résultat final était résumé à travers un portail. Les résultats étaient

encourageants quand le mécanisme était inclus uniquement du côté de l'encodeur. Plusieurs autres approches ont modifié l'architecture de base des Transformer tel que [32], [33] qui ont eu des améliorations intéressantes en termes de BLEU par rapport au modèle Baseline.
Par rapport aux modèles incluant conjointement le contexte de la source et de la cible, [34] fut le premier à expérimenter cette approche. Il l'a fait à partir d'un modèle RNN de base. Cependant, les améliorations n'étaient pas fameuses. Plus récemment [35] a utilisé les multi-encodeurs NMT pour exploiter le contexte des phrases précédentes en combinant les informations de la source et de la cible à travers une concaténation. Ils ont remarqué une amélioration en termes de cohérence et de cohésion lexical. Enfin, nous avons [1] qui est l'un des modèles que nous avons exploré durant notre stage.
Plus récemment, nous avons [2] qui est une méthode que nous avons également étudiée.

3. Notre approche et résultats obtenus

Dans cette partie, nous allons essayer de décrire chronologiquement les expériences que nous avons menées, les pistes que nous avons explorées et les résultats que nous avons obtenus.

3.1 Le HAN

Je devais dans un premier temps implémenter le modèle proposé par [1] en restant le plus près possible du modèle décrit dans l'article et tout en m'adaptant aux contraintes de OpenNMT et de l'environnement de travail.
Une fois le modèle implémenté, l'étape suivante était de le « fine-tune » sur un dataset approprié afin de voir si ce modèle apportait effectivement des améliorations.

3.1.1 Le « fine tune » du HAN

Tout comme les Transformer dont il dépend, le HAN est également très sensible aux paramètres d'optimisation et à la méthode d'entraînement.
En ce qui concerne la méthode, nous avons conservé celle décrite par [1]. Il s'agit d'entraîner le modèle en plusieurs étapes :

- Commencer par entraîner le modèle de traduction phrase après phrase de base. Dans notre cas, il s'agissait du modèle Transformer par défaut tel que décrit par [8], en conservant également les mêmes paramètres d'optimisation.
- Ensuite, il faut greffer la couche HAN sur l'encodeur et continuer l'entraînement jusqu'à convergence. (Base model + HAN_enc)
- Enfin, greffer la HAN sur le décodeur et continuer l'entraînement jusqu'à convergence (base model + HAN_enc-dec).

Cependant, la troisième étape est optionnelle. En effet, même dans l'article original, les améliorations provenant du HAN sur le décodeur n'étaient pas toujours fameuses. D'ailleurs, la plupart des études au niveau de la traduction des documents restent focalisées sur les informations provenant de l'encodeur. Bien que nous avons également mené des expériences avec le HAN_enc-dec, nous nous concentrerons uniquement sur HAN_enc dans la suite de ce rapport.
Lors de la deuxième étape, le choix des paramètres d'optimisation est crucial. En effet, comme le montre la figure 4, deux « learning rate » légèrement différents conduisent à des performances très différentes. Ainsi, comment avons-nous choisi les paramètres d'optimisation ?

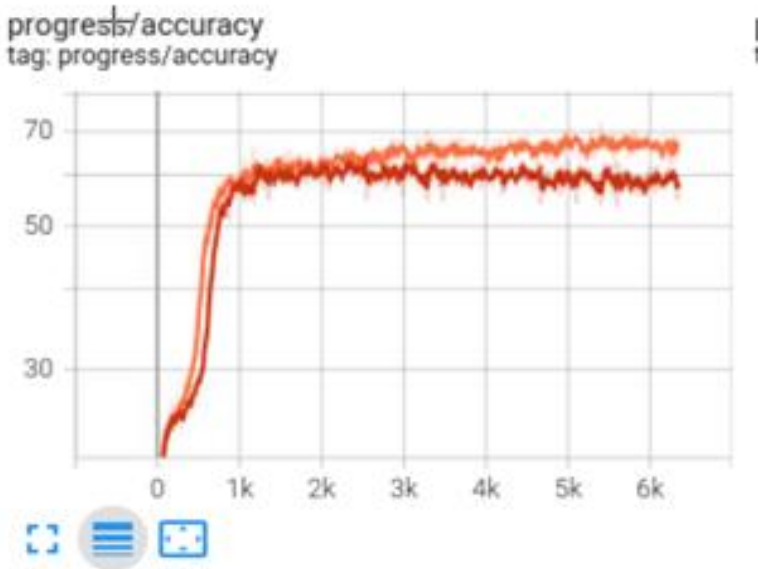

Figure 4: évolution de l'entraînement pour deux learning proche

Nous sommes partis de l'intuition que le learning rate initial devait être assez proche de celui avec lequel l'entraînement de l'étape 1 s'est effectué. Cela empêcherait de trop modifier les paramètres du modèle de base déjà bien entraîné. De plus étant donné que la couche HAN comporte peu de paramètres par rapport au modèle de base, un petit learning rate suffirait à rapidement converger. En outre, on voulait également que le learning rate ne s'éloigne pas trop de son point de départ. C'est pourquoi nous avons choisi un petit « $warmup_{step}$ ». Finalement, la variation du learning rate se faisait suivant la méthode de Noam décrite par [8], suivante :

$$lrate_{step_{num}} = lrate_{step_0} * d_{model}^{-0.5} . \min(step_{num}^{-0.5}, step_{num} . warmup_{step}^{-1.5})$$

Avec $lrate_{step_0}$ de l'ordre de 10^{-1} et $warmup_{step}$ proche de 1.

Ces paramètres nous ont permis à chaque fois d'obtenir les meilleurs résultats. Nous avons testé plusieurs autres paramètres au hasard, un plus grand/petit learning rate,

warmup_step, etc... mais ces paramètres restaient toujours bien inférieure à notre intuition.
Tout ceci nous montre que la méthode d'optimisation de ce type de modèle peut être une piste sérieuse d'amélioration des résultats.
Par rapport à la convergence du modèle, nous avons utilisé la méthode du « early stopping » qui consiste à arrêter l'entraînement dès lors que les performances sur l'ensemble de validation n'évoluent plus. En général, 1 epoch suffisait pour la convergence à l'étape 2 contre 20 à l'étape 1.
Dans la suite de ce rapport, les résultats seront présentés avec les méthodes et paramètres d'optimisations ci-dessus.

3.1.2 Les résultats et analyse sur les petits datasets

Nous avons choisi d'utiliser l'un des datasets proposés par l'article original [1]. Il s'agit du dataset *IWSLT TED 2014 ES-EN*. Il met à notre disposition un ensemble d'entraînement de 0.2M phrases, un ensemble de développement de 2K phrases et 3 ensembles de test (*tst2010-tst2012*) dont la taille concaténée est de l'ordre de 3K phrases. Le tableau 2 récapitule les résultats obtenus :

model	epoch	s-bleu	d-bleu	APT
Base model	20	36	45.7	61.40
Base model +HAN	1	38.1	47.8	61.44

Tableau 2; score HAN sur un petit dataset

Comme annoncé par l'article original, l'évolution en termes de s-BLEU score est assez impressionnante. D'ailleurs, ce score surpasse tous les résultats qui ont été publiés sur ce dataset à notre connaissance et sans utilisation de données supplémentaires.
De plus, l'évolution en termes de d-BLEU est tout aussi intéressante mais elle n'apporte pas beaucoup plus d'informations que le s-BLEU.
Par contre, on remarque que l'évolution du APT est très faible. Rappelons que le APT est une métrique très spécifique à la traduction de document. Ce résultat nous montre que l'amélioration du BLEU score n'est pas forcément dû à une meilleure capacité du modèle à traduire les documents. En effet, [2] dans ses recherches montre que l'amélioration du score de ces modèles hiérarchiques provient parfois d'un surapprentissage lié aux paramètres en plus. En outre, [36] arrive à la même conclusion en montrant que des modèles de traduction au niveau des phrases, bien régularisée arrivent à faire mieux que des modèles de traduction de documents à paramètres et datasets équivalents. On peut également citer [37] qui a montré que les informations provenant du contexte de l'encodeur seraient plus des bruits et que l'amélioration des scores serait dû à un apprentissage robuste.
En effet, le fait que le dataset soit petit favorise ces arguments car il est plus facile de surapprendre sur un dataset petit. De plus [36] a essayé de recenser les phrases où le score en BLEU était meilleure que dans le modèle de bases, il s'est avéré que dans près

de 70 % des cas, l'amélioration en BLEU n'était pas toujours interprétable par des informations contextuelles.
C'est pour toutes ces raisons que nous avons fait une série de tests, pour vérifier si le HAN prenait réellement en compte les informations contextuelles.

3.1.3 Quelques test de performance du modèle HAN

Afin de se rassurer que le HAN était capable de prendre en compte les informations provenant du contexte précédent et non pas juste surapprendre les données, nous avons fait une série de tests dont les plus pertinents sont les suivants :

- Observer la traduction d'un paragraphe exemple dans le dataset et observer les poids d'attention. Le paragraphe en question était :

  Source es: la música es medicina, la música nos cambia.
  Y para Nataniel la música es cordura.
  Porque la música le permite tomar **sus** pensamientos y **sus** delirios y transformarlos a través de **sus** imaginaciones y **su** creatividad en realidad.
  Y esto es un escape de **su** estado atormentado.
  Traduction en: music is medicine, music change us.
  And for Nathaniel, music is mine.
  Because music allows him to take **his** thoughts and **their** delusions and turn through their imaginations and your creativity actually.
  And this is an escape from **your** tormented state.

  Comme on peut le constater, les traductions des pronoms **su** et **sus** dans les deux dernières phrases ne sont pas celles attendues. Etant donné que le sujet est *Nathaniel* dans la deuxième phrase, toutes ces traductions auraient dû être **his**. Cela montre que le modèle n'a pas su tenir compte des informations contenues dans les phrases précédentes. De plus, on a essayé d'observer la contribution de chaque mot des phrases précédentes lors de la traduction du mot **su** dans la dernière phrase. Comme le montre la figure 5, un lien immédiat ne peut être fait entre la traduction de **su** et les poids d'attention. En effet, les mots plus importants (y, para, es, ...) ne sont pas directement liés à la traduction de **su** tandis que le sujet *Nathaniel* a une faible importante. Cela corrobore la piste d'un surapprentissage.

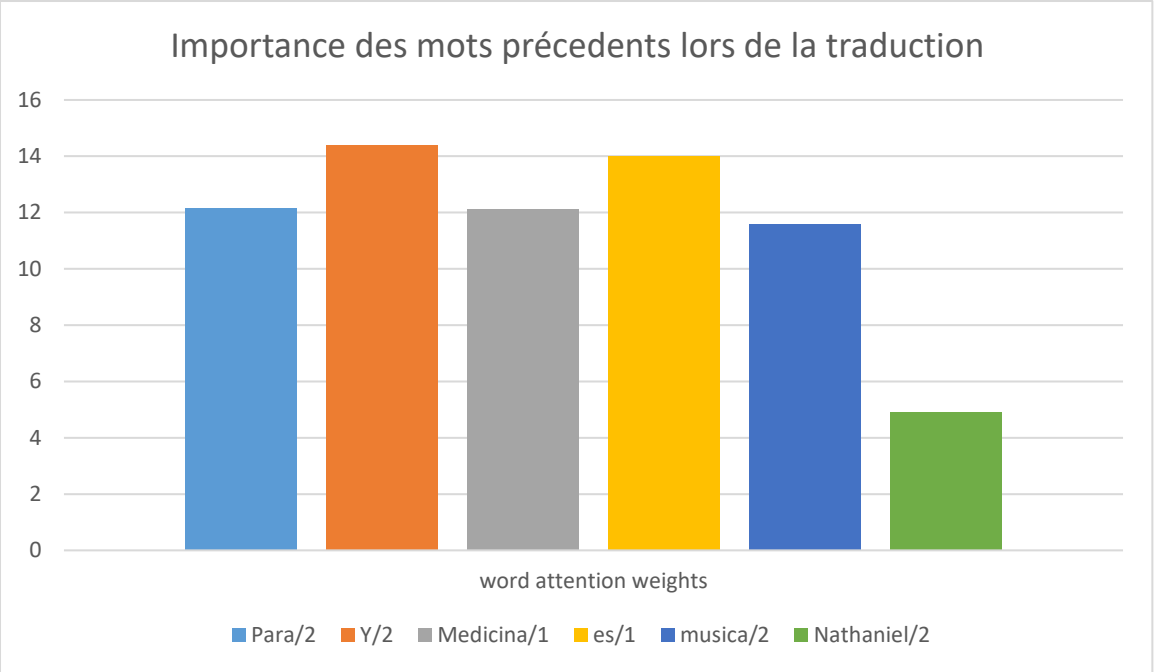

Figure 5: poids d'attention des mots lors de la traduction de su.

- Ensuite, nous avons créé un dataset synthétique constitué uniquement de paragraphes similaires, qui diffèrent uniquement par des variations simples du contexte dans les phrases précédentes. Le but était de forcer le modèle à tenir compte du contexte et de voir s'il en était capable. Ce test a été concluant car le modèle était finalement capable de s'adapter à toutes les variations possibles de contexte dans nos phrases synthétiques. Cela montre que sous certaines conditions, le modèle HAN à une véritable capacité à tenir compte des informations contextuelles.

Ces deux tests nous ont laissé penser qu'entraîner sur un dataset beaucoup plus grand allait permettre d'exploiter le HAN à son plein potentiel car il est difficile de sur-apprendre avec beaucoup de données.

3.1.4 Résultat et analyse sur un gros dataset

Après avoir constitué un gros dataset d'environ 20M avec la méthode décrite en 1.5.1, on a fait les étapes d'entraînement décrit en 3.1.1. Les tests ont été effectués sur le même dataset que dans les parties précédentes à savoir le dataset *IWSLT TED 2014 ES-EN tst2010-tst2012.* Les résultats sont visibles dans le tableau 3.

model	epoch	s-bleu	d-bleu	APT
Base model	4	37.6	46.4	63.23
Base model +HAN	1	38.9	47.9	64.8

Tableau 3:score HAN sur un gros dataset

Cette fois, les résultats sont probants. En effet, on a non seulement une bonne amélioration du BLEU score mais également une amélioration satisfaisante sur la métrique APT. Cela nous montre que le modèle était effectivement capable de tenir compte des phrases du contexte. De plus, une observation de la traduction du paragraphe échantillon précédent nous a montré que le modèle a pu tenir compte du contexte.
En Conclusion, le HAN est un modèle qui marche bien pour la traduction de document quand il est utilisé dans de bonnes conditions. Il faut avoir les bons paramètres d'optimisation et avoir un dataset de qualité et suffisamment grand pour éviter un surapprentissage.

3.1.5 Utiliser HAN sur un modèle NMT déjà existant

Etant donné que mon stage s'inscrivait dans un contexte d'entreprise, il était important que notre modèle puisse servir à l'équipe. C'est pourquoi notre modèle devait pouvoir se greffer aux modèles de traduction déjà existant en production.
D'un point de vue de l'implémentation, la compatibilité était parfaite. Ainsi, on a pu fine tune le modèle *Bnp ES-EN* en production avec le dataset de 20M phrases créé précédemment. Cependant, comme le montre le tableau 4, les améliorations n'ont pas toujours été celles attendues :

model	epoch	s-bleu	d-bleu	APT
Bnp model	?	37.3	46.0	62.39
Bnp model +HAN	1	32.4	41.3	/
Bnp model + N/A	1	32.2	41.1	/

Tableau 4: HAN sur un modèle existant

Comme on peut le constater, les résultats sont plutôt médiocres quand on essaye de greffer le HAN sur un modèle déjà existant. Heureusement, la ligne 3 du tableau nous montre que le problème ne provient pas du HAN mais plutôt d'une différence de distribution entre les données qui ont servi à entraîner le modèle BNP de base et les données qui ont servi pour la fine tune. En effet, en essayant juste de continuer l'entraînement sur le modèle BNP sans HAN, le modèle perd en performance. Cela

montre que le dataset n'est pas compatible avec le modèle BNP de base. Il n'y a donc pas de raison que le HAN fasse beaucoup mieux.
Malheureusement, l'entreprise n'a pas été à même de me fournir le dataset qui avait servi à l'entraînement de leur modèle de base, Le HAN ne pourra donc pas être exploité en production pour le moment.

3.2 Le MR

Face aux lacunes du modèle précédent, notamment sa sensibilité aux paramètres et sa tendance à sur-apprendre, nous avons décidé d'explorer une autre piste de traduction de documents. Nous avons choisi le MR car il a une approche totalement différente du modèle précédent et sa mise en œuvre était simple et rapide. La méthode est décrite plus en profondeur dans la section 1.3.3.

3.2.1 Analyse du MR

Le MR est un modèle qui dépend fortement de la taille maximale des documents dans le dataset qui a servi à l'entraînement. En effet, la taille du nouveau dataset qu'il génère croît d'un facteur d par rapport à la taille initiale. Avec d la taille moyenne des documents. Cependant, compte tenu du temps et des ressources à notre disposition, nous ne pouvions choisir qu'une valeur limitée de d (de l'ordre de 10) même pour les petits datasets tel que *IWSLT TED*. C'est pourquoi la variable MR(d) a été étudiée.
De plus, lors de l'inférence, sa performance dépend également de la taille des blocs qu'elle traduit. Néanmoins, on s'attendait à ce qu'elle face aussi bien sur les petits documents (phrases) que sur les gros documents car dans le dataset généré on a des documents de taille allant de 1 phrase à d phrases. C'est pourquoi nous avons également étudié la variable test-set(d) avec test-set le même ensemble de test que dans les parties précédentes, à savoir *IWSLT TED ES-EN tst2010-2012*.
Le tableau 5 résume les résultats.

model	test(1)		test(3)		*test(6)*		test(9)	
	s-bleu	APT	s-bleu	APT	s-bleu	APT	s-bleu	APT
Base	36	61.40	32.1	60.5	22.4	47.18	11.9	43.18
MR(5)	36.5	62.55	35.8	62.78	25.0	48.85	13.3	43.25
MR(10)	**36.6**	62.82	36.1	**62.88**	25.2	50.62	13.4	43.75

Tableau 5; score MR en fonction de la taille des documents dans l'ensemble de test

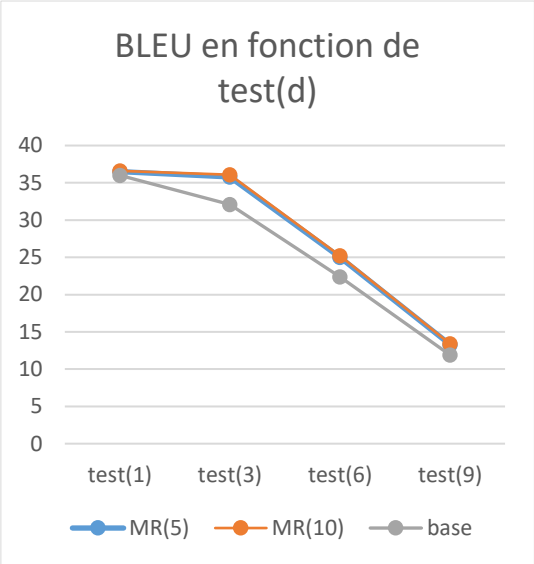

Figure 6: Bleu score selon la taille des documents dans l'ensemble de test

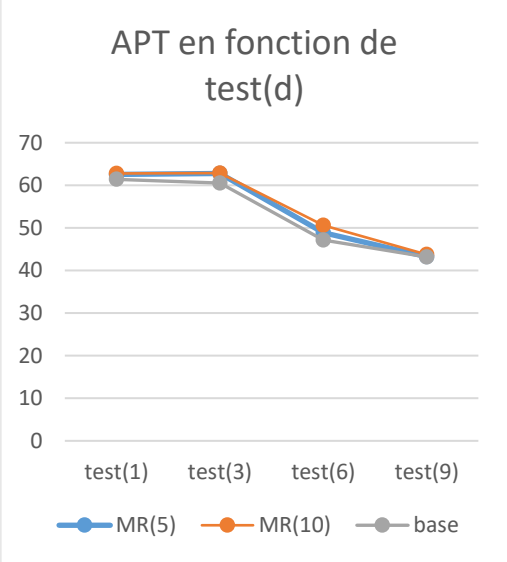

Figure 7:APT score selon la taille des documents dans l'ensemble de test

Comme le montre le tableau ainsi que les deux courbes ci-dessus, le MR fait mieux que le modèle de base que ce soit sur les petits ou longs documents. De plus, on remarque que la performance du MR en termes de BLEU et APT croît avec la taille maximale des documents d dans l'ensemble d'entraînement. Néanmoins, l'écart entre le MR(5) et MR(10) reste relativement faible pour des coûts d'entraînement très différents.
On remarque également que la performance du MR décroît fortement avec la taille des documents dans l'ensemble de test, quand bien même le MR a été entraîné avec une taille de documents supérieure. Cela montre que la traduction de très longues phrases demeure toujours un challenge important. Toutefois, il est important de remarquer que le pic de performance pour la métrique APT est atteint pour test(3), cela montre qu'il existe une taille de document $1 < d'(d) < d$ tel que MR(d) soit le plus performant sur test($d'(d)$).

Il en ressort que, pour exploiter le MR à son plein potentiel, il faut suivre les étapes suivantes :

- Déterminer la taille maximale d des documents dans le dataset en fonction des ressources dont on dispose et en appliquant la formule décrite en 1.3.3
- Evaluer le modèle sur des test-set(d) pour $d \in D \subset [1 \dots d]$ avec D construit aléatoirement
- Repérer la valeur optimale d' telle que le meilleur score pour MR(d) soit obtenu sur test-set(d').

Ayant obtenu les paramètres optimaux du MR, on a pu le confronter au HAN.

3.2.2 MR vs HAN

Les deux modèles ont été entraîné avec les mêmes datasets et en utilisant les meilleurs paramètres trouvés pour chacun d'eux. Le tableau ci-dessous permet de visualiser les résultats :

model	s-bleu	d-bleu	APT
Base model	36.0	45.7	61.40
Base model +HAN	**38.1**	47.8	61.44
MR(10)	36.6	46.6	**62.88**

Tableau 6: HAN vs MR

Ce tableau nous montre que le meilleur BLEU est atteint par le HAN tandis que le meilleur APT est atteint par le MR. Cela permet de confirmer l'hypothèse selon laquelle l'amélioration en termes de BLEU du HAN ne provient pas toujours des informations liées aux documents. Cela nous montre également la capacité du MR à mieux traduire les documents sans toutefois être meilleur dans l'ensemble. Néanmoins, les deux modèles traduisent mieux que le modèle de base, que ce soit au niveau des phrases ou des documents, cela laisse un éventail de choix pour l'entreprise, si elle souhaite améliorer sa traduction au niveau des documents.

3.3 D'autres pistes

Tout au long de notre stage, nous avons eu plusieurs autres idées. D'autres ont été testées et d'autres sont laissées pour d'éventuelles futures recherches. On peut citer entre autres :
Le Deep Han : L'idée était de superposer plusieurs couches HAN les unes au-dessus des autres dans le but d'avoir un apprentissage hiérarchique profond. Cette piste a été testée mais les résultats n'ont pas été concluants.
L'optimisateur : Comme décrit dans la partie 3.1.1, le HAN était très sensible à la variation des paramètres. Nous avons choisi nos paramètres sur la base d'une intuition.

Bien que nos résultats étaient meilleurs avec ces paramètres que tous les autres que nous avons eu à tester, nous pensons que se concentrer sur la méthode et les paramètres d'optimisation peut être une piste d'amélioration.
MR sur un gros dataset : Nous n'avons pu évaluer le MR que sur des petits datasets à cause de l'augmentation en temps qu'elle impliquait. Cependant, toute solution étant destinée à être en production, trouver des moyens de l'exploiter sur des plus gros datasets avec les mêmes ressources pourraient s'avérer très intéressant.
HAN+MR : partant du constat que HAN était plutôt complémentaire au MR, on s'est tout de suite demandé si nous ne pourrions pas combiner les deux. C'est-à-dire utiliser le HAN avec le dataset générer par le MR. Nous n'avons malheureusement pas eu assez de temps pour tester cette idée.

Conclusion

Tout au long de notre stage, nous avions pour mission d'étudier la traduction au niveau des documents. Après une revue de la bibliographie, nous avons principalement exploré deux pistes. Le HAN et le MR. Les deux ont pu être implémentées et testées sous plusieurs facettes au sein de l'entreprise.
Il en ressort que, ces deux modèles apportaient une réelle amélioration en termes de métrique générale de traduction tel que le BLEU ainsi que sur des métriques spécifiques aux documents tel que le APT.
Cependant, chacune de ces méthodes présentait des lacunes telles qu'une grande sensibilité aux paramètres de la part du HAN et un coût en mémoire élevé pour le MR. Ces problèmes rendent difficile une mise en production de ces modèles à l'état actuel.
Néanmoins, les résultats obtenus ouvrent de sérieux champs de recherche pour la traduction au niveau des documents. En effet, la confrontation entre le MR et le HAN nous a montré que les solutions qui vectorisent globalement les documents sont les plus prometteuses pour la traduction de document mais reste très couteuses en mémoire. Ainsi, trouver des moyens d'exploiter ces modèles avec des ressources abordables serait une piste très intéressante de recherche.

References

[1] L. a. R. D. a. M. a. P. N. a. H. J. Miculicich, "Document-Level Neural Machine Translation with Hierarchical Attention Networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[2] Z. a. W. M. a. Z. H. a. Z. C. a. H. S. a. C. J. a. L. L. Sun, "Rethinking Document-level Neural Machine Translation," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 3537--3548.

[3] M. a. A.-O. Y. Freitag, "Beam search strategies for neural machine translation," *arXiv preprint arXiv:1702.01806,* 2017.

[4] P. a. H. H. a. B. A. a. C.-B. C. a. F. M. a. B. N. a. C. B. a. S. W. a. M. C. a. Z. R. a. o. Koehn, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, 2007, pp. 177--180.

[5] Y. a. K. T. a. F. S. a. T. M. a. S. A. a. S. T. a. A. S. Shibata, "Byte Pair encoding: A text compression scheme that accelerates pattern matching," Citeseer, 1999.

[6] T. a. C. K. a. C. G. a. D. J. Mikolov, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781,* 2013.

[7] J. a. C. M.-W. a. L. K. a. T. K. Devlin, "Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018.

[8] A. a. S. N. a. P. N. a. U. J. a. J. L. a. G. A. N. a. K. {. a. P. I. Vaswani, "Attention is all you need," *Advances in neural information processing systems,* vol. 30, 2017.

[9] J. a. L. H. a. S. M. a. Z. F. a. X. J. a. Z. M. a. L. Y. Zhang, "Improving the transformer translation model with document-level context," *arXiv preprint arXiv:1810.03581,* 2018.

[10] Y. a. G. J. a. G. N. a. L. X. a. E. S. a. G. M. a. L. M. a. Z. L. Liu, "Multilingual denoising pre-training for neural machine translation," *Transactions of the Association for Computational Linguistics,* vol. 8, pp. 726--742, 2020.

[11] K. a. R. S. a. W. T. a. Z. W.-J. Papineni, "Bleu: a method for automatic evaluation of machine translation," in *{Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311--318.

[12] M. Post, "A Call for Clarity in Reporting {BLEU} Scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels, Association for Computational Linguistics, 2018, pp. 186--191.

[13] L. M. a. P.-B. A. Werlen, "Validation of an automatic metric for the accuracy of pronoun translation (APT)," in *Proceedings of the Third Workshop on Discourse in Machine Translation*, 2017, pp. 17--25.

[14] Y. a. M. S. a. Z. D. a. Y. J. a. H. H. a. Z. M. Jiang, "BlonD: An Automatic Evaluation Metric for Document-level MachineTranslation," *arXiv preprint arXiv:2103.11878,* 2021.

[15] C. a. C. V. a. S. N. A. Dyer, "A simple, fast, and effective reparameterization of ibm model 2," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 644--648.

[16] B. T. a. K. C. Wong, "Extending machine translation evaluation metrics with lexical cohesion to document level," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 1060--1068.

[17] M. a. N. J. a. S. S. a. B. L. a. F. M. Cettolo, "Report on the 11th IWSLT evaluation campaign," in *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, 2014.

[18] G. a. K. Y. a. D. Y. a. S. J. a. R. A. M. Klein, "Opennmt: Open-source toolkit for neural machine translation," *arXiv preprint arXiv:1701.02810,* 2017.

[19] P. F. a. C. J. a. D. P. S. A. a. D. P. V. J. a. J. F. a. M. R. L. a. R. P. Brown, "A statistical approach to language translation," in *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*, 1988.

[20] D. a. C. K. a. B. Y. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473,* p. 2014.

[21] Z. a. L. Z. a. L. Y. a. L. X. a. L. H. Tu, "Modeling coverage for neural machine translation," *arXiv preprint arXiv:1601.04811,* 2016.

[22] D. a. L. F. Coldewey, "DeepL schools other online translators with clever machine learning," *TechCrunch. Abgerufen,* p. 2020, 2017.

[23] E. Wdowiak, "Sicilian Translator: A Recipe for Low-Resource NMT," *arXiv preprint arXiv:2110.01938,* 2021.

[24] X. a. D. K. a. L. L. a. G. J. Liu, "Very deep transformers for neural machine translation," *arXiv preprint arXiv:2008.07772,* 2020.

[25] S. a. O. M. a. A. M. a. G. D. Edunov, "Understanding back-translation at scale," *arXiv preprint arXiv:1808.09381,* 2018.

[26] P.-t. m. n. m. t. b. l. a. information, "Pre-training multilingual neural machine translation by leveraging alignment information," *Pre-training multilingual neural machine translation by leveraging alignment information,* 2020.

[27] L. a. L. J. a. W. Y. a. M. Q. a. Q. T. a. C. W. a. Z. M. a. L. T.-Y. a. o. Wu, "R-drop: Regularized dropout for neural networks," *Advances in Neural Information Processing Systems,* vol. 34, pp. 10890--10905, 2021.

[28] V. d. t. f. n. m. translation, "Very deep transformers for neural machine translation," *arXiv preprint arXiv:2008.07772,* 2020.

[29] S. L. O. F. a. K. C. 2. D. N. M. T. B. Sebastien Jean, " Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does Neural Machine Translation Benefit," *arXiv preprint arXiv:1704.0513,* 2017.

[30] L. a. T. Z. a. W. A. a. L. Q. Wang, "Exploiting cross-sentence context for neural machine translation," *Wang, Longyue and Tu, Zhaopeng and Way, Andy and Liu, Qun,* 2017.

[31] E. a. S. P. a. S. R. a. T. I. Voita, "Context-aware neural machine translation learns anaphora resolution," *arXiv preprint arXiv:1805.10163,* 2018.

[32] K. a. M. S. a. H. G. Wong, "Contextual neural machine translation improves translation of cataphoric pronouns," *arXiv preprint arXiv:2004.09894,* 2020.

[33] J. a. L. H. a. S. M. a. Z. F. a. X. J. a. Z. M. a. L. Y. Zhang, "Improving the transformer translation model with document-level context," *Zhang, Jiacheng and Luan, Huanbo and Sun, Maosong and Zhai, Feifei and Xu, Jingfang and Zhang, Min and Liu, Yang,* 2018.

[34] J. a. S. Y. Tiedemann, "Neural machine translation with extended context," *Tiedemann, J{\"o}rg and Scherrer, Yves,* 2017.

[35] R. a. S. R. a. B. A. a. H. B. Bawden, "Evaluating discourse phenomena in neural machine translation," *Bawden, Rachel and Sennrich, Rico and Birch, Alexandra and Haddow, Barry,* 2017.

[36] Y. a. T. D. T. a. N. H. Kim, "When and why is document-level context useful in neural machine translation?," *arXiv preprint arXiv:1910.00294,* 2019.

[37] B. a. L. H. a. W. Z. a. J. Y. a. X. T. a. Z. J. a. L. T. a. L. C. Li, "Does multi-encoder help? A case study on context-aware neural machine translation," *arXiv preprint arXiv:2005.03393,* 2020.

[38] D. a. A. B. a. L. W. a. G. L. a. W. Y. Guthrie, "A closer look at skip-gram modelling," in *Proceedings of the fifth international conference on language resources and evaluation (LREC'06)*, 2006.