Institut National des Langues et Civilisations Orientales
Département Textes, Informatique, Multilinguisme

Adaptation des systèmes de traduction automatique neuronale aux domaines spécialisés

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :
Ingénierie Multilingue
par

Yunbei ZHANG

Directeur de mémoire :
Cyril Grouin

Encadrante :
Anabel Rebollo

Année universitaire 2017/2018

Table des matières

Liste des figures

Liste des tableaux

Remerciements

Tout d'abord j'adresse mes sincères remerciements à mon maître de stage, Mme. Anabel Rebollo, responsable du service du traitement automatique de la langue au sein de l'entreprise Systran, qui est toujours disponible pour mes questions et m'a aidé tout au long de la période de stage.

Je tiens à remercier également Guersande Chaminade, Dakun Zhang, Josep Crego, Elise Michon et Chiyo Geiss pour le temps qu'ils ont consacré, leur support, et en particulier pour leur aide dans l'expérimentation et la relecture de mon mémoire.

J'adresse mes plus sincères remerciements à mon encadrant universitaire, Mr. Cyril Grouin, pour sa disponibilité et conseils précieux ainsi que ses encouragements tout au long de la rédaction de ce mémoire.

Résumé

La traduction automatique neuronale est une technique émergente dans la discipline de linguistique informatique. L'entraînement du modèle de traduction neuronale est basé sur un corpus parallèle. Traduire des textes d'un domaine non représenté dans le corpus d'entraînement s'avère difficile et produit une qualité de traduction peu satisfaisante. Ce travail de recherche a été effectué à l'aide du système de traduction automatique neuronale implémenté par Systran. Il s'agit d'entraîner des modèles de traduction français-chinois à travers un processus de spécialisation en fine-tuning, et également des modèles de classification automatique de texte. Nous avons investigué une méthode hybride qui consiste à calculer pour chaque phrase du document à traduire, la probabilité qu'elle appartienne à chaque classe prédéfinie. La probabilité sera considérée comme un poids sur le score de confiance assigné sur chaque phrase de traduction générée par le système de traduction, et la traduction recueillant un meilleur score pondéré sera sélectionnée et réécrite dans un nouveau fichier de sortie. La traduction sera accumulée phrase par phrase dans ce fichier de sortie en construisant une traduction synthétique.

La tâche de classification automatique de texte a été réalisée avec l'algorithme de Ngramme et Naïve Bayes qui nous permettent d'avoir une meilleure F-mesure (100%). L'adaptation du système de traduction aux domaines spécialisés améliore le score BLEU.

Mots-clés : Adaptation des domaines, traduction automatique neuronale, méthode hybride, classification multi-classe, Ngramme, Naïve Bayes, perplexité, valeur pondérée, traduction synthétique, F-mesure, BLEU

Introduction

Avant-propos

Avec le développement des nouvelles technologies, nous nous retrouvons confrontés à l'impact de l'intelligence artificielle. Il y a deux ans, le système AlphaGo a remporté le jeu de go, et pour la première fois, une machine a gagné une partie contre un joueur de go légendaire. En juin 2018, un système d'intelligence artificielle appelé Project Debater, créé par IBM et destiné à débattre avec des humains, a gagné dans les débats contre deux débatteurs chevronnés. Du fait des grands progrès et succès, nous sommes amenés à une nouvelle ère de l'intelligence artificielle. En tant que champ interdisciplinaire, le traitement automatique des langues (Natural language processing, NLP) dépend à la fois de l'intelligence artificielle et des sciences cognitives. Le TAL vise à découvrir les règles des langues naturelles et à développer des modèles de sorte que la machine puisse traiter les langues naturelles comme un humain. Il se compose de plusieurs sous-disciplines telles que la reconnaissance automatique de la parole, la recherche d'information et la traduction automatique. Ce dernier domaine constitue un domaine de recherche important qui a pour objectif de traduire un texte d'une langue à l'autre.

La langue est un outil essentiel de communication et réflexion humaine et joue un rôle indispensable dans la transmission de l'information. La variété des langues reflète les différences de cultures et d'origines, qui peut entraver une communication interlangue. Pourtant cette contrainte a généré les professions d'interprètes et de traducteurs. Aujourd'hui, nous sommes aussi à l'ère du Big data. «Selon Netcraft Ltd[1], il y avait 717 millions de sites Web en août 2013 contre 176 millions fin avril 2008. L'accélération du nombre de sites dans des langues multiples rend l'utilisation de la traduction automatique indispensable. Sans les logiciels de traduction automatique, la majorité des sites ne serait jamais traduite» [***ref. Systran***[2]]. En 2016, Google présente un système de traduction de haute qualité, GNMT [***Wu et al., 2016***], démontrant une capacité de traduction proche de celle de l'humain dans certaines paires de langues. À partir de là, le niveau moyen de traduction automatique a fait un progrès qualitatif.

Systran

Systran est le leader mondial des technologies de traduction automatique, et commercialise des solutions et produits innovants facilitant la communication multilingue des entreprises et des particuliers avec la possibilité de faciliter la communication dans plus de 130 combinaisons de langues. Plusieurs produits sont proposés dans le but d'améliorer

1Netcraft est une entreprise basée à Bath en Angleterre. Elle est spécialisée dans les technologies Internet.
2http://www.systran.fr/systran/technologie/pourquoi-la-traduction-automatique/

l'efficacité et la productivité des entreprises dans de nombreux domaines : le eCommerce, le support client ou la gestion des connaissances.

- SystraNet (service gratuit de traduction en ligne).
- SystranLinks (service de traduction de sites Web).
- Gadget Systran : traducteur et dictionnaire en ligne pour les utilisateurs Windows.
- Plusieurs versions Desktop ou Pocket PC
- Une version serveur

Cadre du stage et présentation du sujet

Les travaux de recherche présentés dans ce mémoire ont été effectués à l'occasion d'un stage à Systran SA à Paris, au sein de l'équipe d'ingénierie de traitement du langage naturel. Dans le cadre du projet PN9, qui a pour but l'implémentation d'une nouvelle génération de logiciel de traduction avec la technologie de l'état de l'art. Ce stage s'inscrit dans une tentative de la consolidation de processus automatiques d'analyse, et tout particulièrement de la génération des modèles de traduction neuronaux et l'évaluation de la traduction prédite.

En traduction, la langue source est la langue de départ et la langue cible est la langue d'arrivée. À la différence de la traduction traditionnelle basée sur des phrases, un système de traduction neuronale peut être entraîné directement à partir du texte source et cible, il prend une phrase en entrée et génère une traduction en sortie. Dans un système de bout en bout (end-to-end), un seul modèle est utilisé pour générer une traduction. Il nécessite un prétraitement et la constitution de ressources linguistiques afin de générer un modèle de traduction.

Dans une perspective pragmatique, un modèle de traduction peut être générique ou spécialisé selon les besoins du client. Il permet de limiter la portée des vocabulaires dans certains domaines spécifiques. L'adaptation de la traduction à un domaine spécialisé tel que le tourisme, technique ou journalistique est nécessaire pour les entreprises et organisations mondiales. Les études présentées dans ce mémoire ont démarré dans ce contexte en visant à explorer une nouvelle possibilité d'amélioration du résultat de traduction.

Nous présentons dans un premier temps l'état de l'art des technologies de la traduction automatique, ainsi que des travaux similaires. En second lieu, nous passons à la phase de préparation des matériels expérimentaux et à la méthodologie. Le troisième chapitre présente la partie des expérimentations. Enfin, le dernier chapitre se compose de l'analyse des résultats des expériences et la conclusion.

Définitions

Nous définissons ci-dessous quelques notions techniques utilisées dans le mémoire :

- NLP (Natural language processing) : Traitement automatique des langues
- MT (Machine Translation) : Traduction automatique
- GNMT : Google Neural Machine Translation
- NMT (Neural machine translation) : Traduction automatique neuronale
- RBMT (Rule-based machine translation) : Traduction automatique à base de règles
- SMT (Statistical machine translation) : Traduction automatique statistique
- DA (Domain adaptation): Adaptation au domaine
- MPR (monolingual preprocessing) : Prétraitement monolingue
- BPR (bilingual preprocessing) : Prétraitement bilingue
- OOV (out-of-vocabulary) : Mot en dehors du vocabulaire prédéfini
- Époque : Nombre de cycles d'apprentissage
- Batch : Nombre de données apprises simultanément
- BLEU (BiLingual Evaluation Understudy) : Algorithme d'évaluation de la qualité du texte qui a été traduit automatiquement d'une langue naturelle à une autre
- LM (language model) : Modèle de langue
- Score de confiance : Seuil qui détermine le score correspondant le plus bas acceptable
- Perplexité : Mesure de la granularité avec laquelle un modèle accomplit sa tâche

1. État de l'art

Sommaires

1.1 Introduction

Ce chapitre présente d'abord un bref aperçu de l'évolution de la traduction automatique, ainsi que l'état de l'art dans ce domaine. Ensuite nous serons amenés à examiner différentes approches de spécialisation.

1.2 Origine et développement de la traduction automatique en bref

Généralement il existe 3 grandes approches de la traduction automatique (Machine translation, MT) en fonction de différentes phases historiques : traduction automatique à base de règles (Rule-Based Machine Translation, RBMT), traduction statistique (Statistical Machine Translation, SMT) et traduction neuronale (Neural Machine Translation, NMT).

Le début de traduction automatique remonte aux années 1950. L'expérience de Georgetown [***Georgetown University and IBM*, 1954**] a connu un succès qui consiste à une traduction par IBM 701[3] d'une quarantaine phrases du russe vers l'anglais. Cependant, cette technologie n'a pas été développée davantage à cause du niveau de l'informatique n'était pas assez développé. Dès les années 1970, les premiers systèmes de traduction automatique à base de règles (RBMT) ont été développés parmi lesquels Systran, Japanese MT systems[4] et EUROTRA[5]. Dans cette approche, en principe, les mots sont traduits de manière linguistique en employant des règles linguistiques et des entrées dictionnaires. En réalité, le nombre de langues est fini alors que celui des règles linguistiques est infini, ce qui se révèle un obstacle insoluble. Cette problématique a été résolue par l'émergence de la traduction automatique statistique (SMT) au début des années 1990, lorsque le centre de recherche IBM a introduit leurs résultats de recherche consistant en 5 modèles de SMT [***P. Brown et al., 1993***]. Cette approche est bâtie sur la théorie de l'information. Théoriquement,

3Le premier ordinateur commercialisé par la compagnie IBM
4http://aamt.info/english/mtsys.htm
5https://en.wikipedia.org/wiki/Eurotra

un document est traduit en fonction de la distribution de probabilité qu'une chaîne de caractères dans la langue cible soit la traduction d'une autre chaîne de caractères dans la langue source. La SMT est une grande évolution qui a mis en œuvre une simplification des processus de traduction où les règles linguistiques et dictionnaires ne sont plus nécessaires. Cette solution montre aussi des difficultés telles que sa faible performance dans la traduction d'une paire de langues de parenté différente du fait de sa faiblesse en traitement de la syntaxe et sémantique : ainsi, malgré une traduction correcte pour chaque mot, la structure syntaxique de la traduction n'est pas toujours naturelle. L'idée d'introduction des réseaux neuronaux dans la MT est apparue en 2014, à l'Université de Montréal [***Cho, Bengio et al., 2014***]. En 2016, la MT a inauguré une nouvelle ère. Google [***Wu et al., 2016***] a présenté dans des travaux de recherche impressionnants, Google Neural Machine Translation (GNMT), un système qui offre une traduction pour 8 langues et basé sur l'apprentissage profond (Deep Learning). Le résultat de l'expérience montre que la GNMT permet de réduire les erreurs de traduction de 55% à 85% dans plusieurs paires de langues.

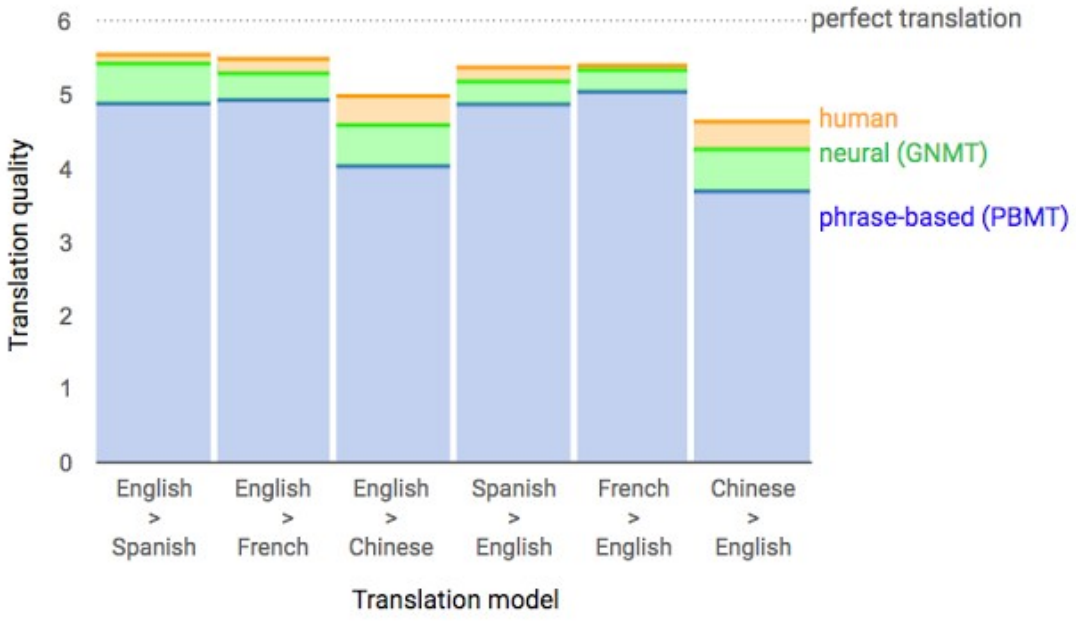

FIGURE 1.1 Qualité de traduction par des intermédiaires différents

Notes : Figure provient de l'article de Google AI Blog[6]

1.3 La traduction automatique neuronale (NMT) et son architecture

La NMT est différente des méthodes traditionnelles en ce qu'elle modélise une phrase entière. Depuis quelques années, la technique de réseaux de neurones récurrents (RNN) a été mise en pratique dans la MT pour réaliser une traduction de bout en bout. Un

6https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

système de traduction automatique à base de RNN a deux composantes élémentaires : un réseau encodeur (encoder) et un réseau décodeur (decoder). L'encodeur lit une phrase dans la langue source en entrée et la transforme en une représentation vectorielle (un vecteur contextuel) de longueur fixe, ensuite à partir de ce vecteur, la traduction en langue cible est générée mot par mot par le décodeur. Bien que l'architecture de l'encodeur-décodeur s'avère très efficace, elle est moins performante pour traiter une séquence de mots de grande longueur en langue source. Au fur et à mesure que la phrase s'allonge, les informations sémantiques à capturer sont de plus en plus volumineuses et il est de plus en plus difficile pour l'encodeur d'assembler toutes ces informations en un seul vecteur de longueur fixe à partir duquel le décodeur va générer une traduction en langue cible, notamment pour les phrases qui sont plus longues que les phrases du corpus de l'entraînement [***Cho et al., 2014***].

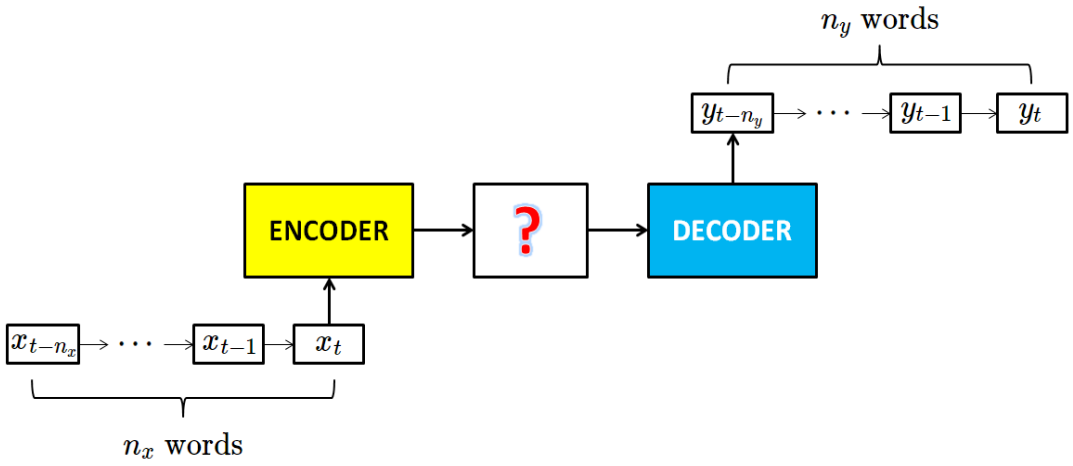

FIGURE 1.2 Architecture d'encodeur-décodeur

Notes : Figure provient du site https://chunml.github.io/ChunML.github.io/project/Sequence-To-Sequence/

Cette contrainte a été partiellement résolue avec l'application d'un mécanisme d'attention additionnel dans le domaine de la MT [***Bahdanau et al., 2015***]. Cette technique s'inspire du mécanisme d'attention visuelle des êtres humains, qui est un mécanisme de traitement des signaux cérébraux unique à la vision humaine. En balayant rapidement l'image globale, la vision humaine identifie la zone cible sur laquelle l'attention doit être focalisée, puis investit davantage d'attention dans cette zone pour obtenir des informations plus détaillées et ignorer d'autres informations non utiles, ce qui améliore à la fois l'efficacité et la précision du traitement de l'information visuelle. De la même façon, le mécanisme d'attention introduit dans la MT rend l'encodage préliminaire d'une phrase complète en un vecteur fixe non nécessaire. Il permet au modèle d'apprendre où placer l'attention sur la phrase d'entrée lorsque le décodeur produit chaque mot de la traduction.

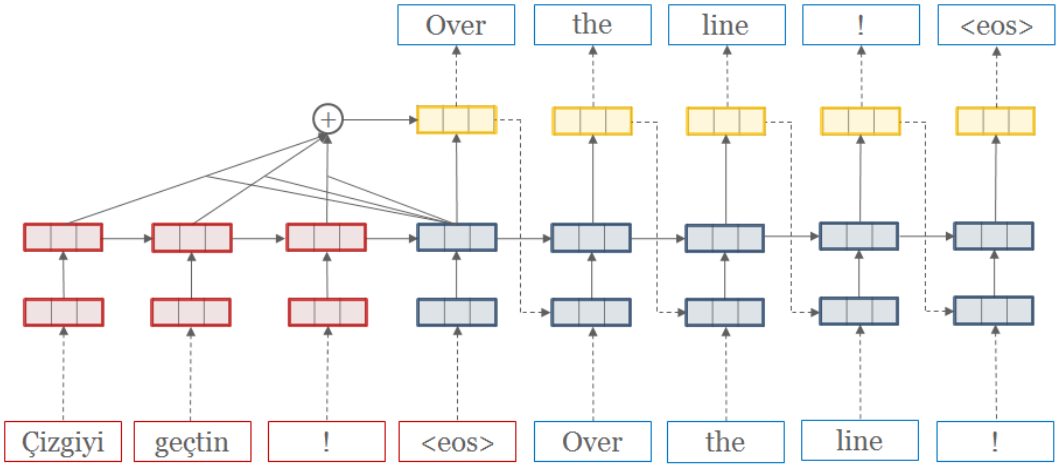

FIGURE 1.3 Architecture de la NMT

Notes : Figure tirée de la démonstration sur le site Opennmt

Bien que l'architecture de RNN encodeur-décodeur (avec le mécanisme d'attention) permet d'augmenter la qualité de traduction, l'entraînement des modèles de traduction nécessite énormément de temps. En conséquence, Google [***Vaswani et al., 2017***] a proposé une nouvelle architecture de réseau, le Transformer, basée uniquement sur des mécanismes d'attention. Le Transformer ne nécessite aucune récursivité, il traite en parallèle tous les tokens d'une séquence et joint les tokens distants par le mécanisme de self-attention. Les expériences internes et externes à Google montrent que le Transformer surpasse les meilleurs résultats existants, ce qui l'établit comme état de l'art de la NMT.

1.4 Travaux similaires

L'adaptation au domaine (Domain Adaptation, DA) est une thématique récente dans la NMT, alors qu'elle a été largement utilisée dans la SMT et a prouvé son efficacité. L'adaptation au domaine intégrée dans la SMT peut être groupée en 5 catégories [***Boxing Chen et al., 2017***] : par auto-entraînement, à base du contexte, à base de la thématique, par sélection des données et par pondération sur les données. L'approche de l'auto-entraînement est étroitement liée à la technique de traduction inverse (back-translation). Dans ce mémoire de recherche, nous nous détaillerons l'approche de DA par sélection des données et pondération sur ces dernières. La DA s'oppose à un système de traduction générique : elle consiste à entraîner un modèle de traduction à partir de corpus dans des catégories spécialisées, tel que des documents concernant l'informatique, le tourisme, le discours oral, etc.

La notion de spécialisation en NMT a d'abord été introduite par les travaux de Systran [***Servan et al., 2016***]. De manière différente aux méthodes traditionnelles, cette approche adapte un modèle générique à un domaine spécialisé par son entraînement pendant plusieurs itérations supplémentaires sur des données du domaine. L'entraînement

du modèle en faisant une spécialisation permet non seulement d'améliorer la qualité de traduction, mais permet aussi de gagner en vitesse par rapport à d'autres méthodes de DA.

DA peuvent s'appliquer à une phrase comme elles peuvent s'appliquer à un document ou un corpus. L'approche de DA par pondération sur les données consiste à pondérer chaque unité de données suivant sa similarité aux données du domaine. L'approche de pondération des coûts (cost-weighting) a été présentée initialement par les membres du CNRC[7] [***Boxing Chen et al., 2017***]. Dans cette approche, la pondération des coûts s'applique sur chaque phrase et a été réalisée en implémentant conjointement un classifieur du domaine et un modèle de NMT. Le classifieur permet de prédire le domaine d'une phrase et y attribue une probabilité par domaine. De ce fait, le modèle de traduction attache plus d'importance à des données du domaine. Pour la tâche de traduction du chinois en anglais, le résultat a surpassé leur baseline de 1,2 point BLEU, et 0,8 pour la traduction de l'anglais en chinois.

Nos études poursuivent l'approche de spécialisation par pondération sur les données. Nous cherchons une nouvelle possibilité de DA basée sur deux unités de données, un document et chaque phrase dans ce document par ajustement des coûts. Nous abordons en détail notre approche dans le chapitre 2 (Méthodologie).

7Conseil national de recherches Canada

2. Méthodologie

Sommaires

2.1 Introduction

Dans ce chapitre nous présentons d'abord le système de NMT implémenté par Systran et sa composition. Dans un second lieu, nous introduisons les méthodes que nous utilisons pour l'expérimentation. Nos méthodes se décomposent en trois phases. Dans la première phase, il s'agit de générer des modèles de traduction génériques et spécialisés. Dans la deuxième phase nous présentons les algorithmes de similarité choisis pour entraîner des modèles de classification, nous envisageons de prédire les domaines des documents du corpus de test ainsi que le domaine de chaque phrase dans le document. Enfin, dans la dernière phase nous allons associer une traduction pertinente au document en fonction de la pondération calculée.

2.2 Système de traduction automatique neuronale à Systran

Aujourd'hui, le noyau de Systran est son outil de traduction automatique utilisant un réseau de neurones, contrairement aux versions précédentes, dans lesquelles l'outil exploitait un ensemble de règles linguistiques (dictionnaires composés de millions d'entrées, mais également données syntaxiques, morphologiques et sémantiques) et effectuait un traitement statistique.

En parcourant le texte source, le logiciel crée une représentation intermédiaire à partir de laquelle la traduction est générée. Le logiciel n'utilise alors plus de règles pour transférer la structure grammaticale du texte source au texte cible.

Contrairement aux technologies jusqu'alors utilisées sur le marché (statistiques ou autres), l'entreprise entraîne depuis un an et demi des modèles de NMT. Un moteur traite la totalité du processus de traduction au moyen d'un unique réseau de neurones artificiels.

Cette technologie qui se base sur des algorithmes complexes à la pointe du Deep Learning permet au moteur PNMT™ (Pure Neural™ Machine Translation) d'apprendre et de généraliser les règles d'une langue à partir d'une traduction de référence et de produire une traduction de bonne qualité.

En 2018, Systran lance la nouvelle version de son produit (PN9), qui a pour le but de passer d'une configuration statique (version 8) à un processus dynamique avec un cycle d'entraînement infini et une architecture à la fois légère et robuste.

Le cœur du PN9 est le système OpenNMT, qui est un système de traduction neuronale Open Source, développé par SYSTRAN et Harvard[8] et figurant parmi les deux projets finalistes lors de WMT[9] en 2017 [***Klein et al, 2018***]. Il est lancé en décembre 2016, et compte déjà plusieurs centaines d'utilisateurs et de contributeurs, issus du monde académique et industriel. Une vingtaine de chercheurs, linguistes et ingénieurs du centre R&D de SYSTRAN SA, basé à Paris, travaillent au développement de cette plateforme et interagissent avec la communauté d'utilisateurs.

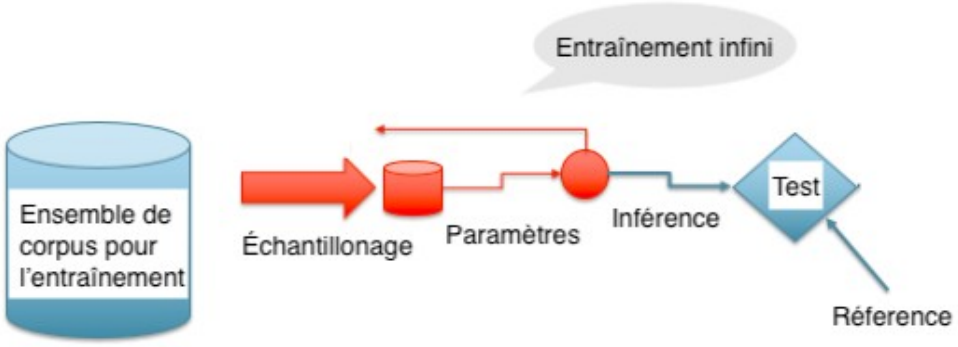

FIGURE 2.1 Flux de travail du système PN9

8http://nlp.seas.harvard.edu
9Workshop on Machine Translation http://www.statmt.org/wmt17/

Comme toutes les tâches de MT, l'entraînement d'un modèle nécessite un corpus parallèle (ou un corpus comparable pour des paires de langues pour lesquelles nous trouvons très peu de ressources), un prétraitement sur les données provenant du corpus et éventuellement un post-traitement sur la traduction générée.

Un fichier de configuration au format json incluant toutes les informations sur les ressources (configuration pour l'échantillonnage des données, prétraitement, tokenisation et post-traitement) ainsi que des paramètres pour l'entraînement est intégré dans le système. Il est appliqué tout au long du processus afin de faire des ajustements instantanément. Nous pouvons mettre à jour les paramètres à l'intérieur de ce fichier au début de chaque cycle de l'entraînement.

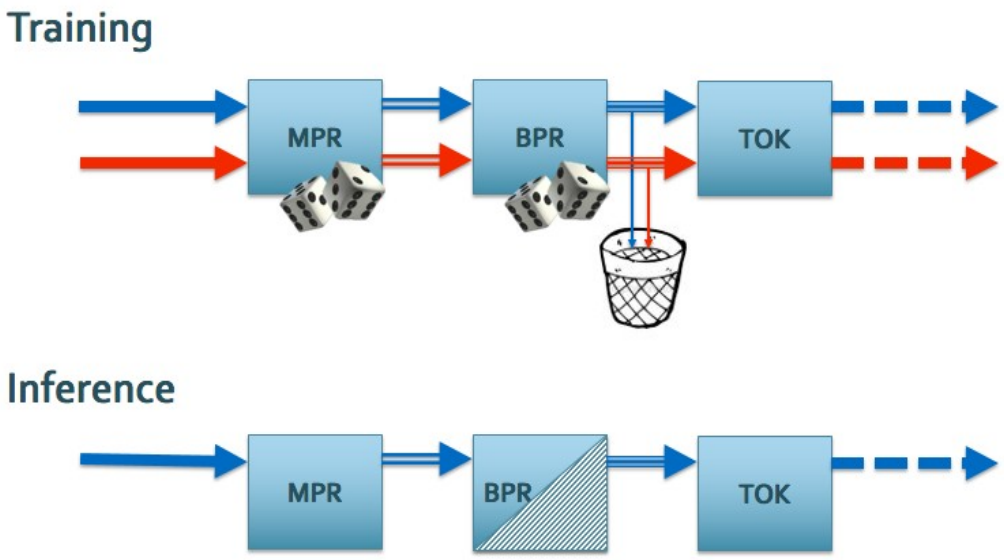

FIGURE 2.2 Processus de l'entraînement infini
(MPR : Prétraitement monolingue, BPR : Prétraitement bilingue, TOK : Tokenisation)

Le processus est basé sur la préparation des données bilingues de façon dynamique, qui se reflète dans l'échantillonnage de données et le prétraitement.

2.2.1 Échantillonnage des données (Data sampling)

Un corpus est un ensemble de documents et un corpus parallèle est un ensemble de textes en langue source alignés avec leur traduction en langue cible. Les données bilingues pour un système de traduction sont généralement issues du corpus parallèle avec des phrases alignées. Cependant pour certaines paires de langues, les corpus parallèles sont seulement disponibles pour un nombre limité de domaines. Les corpus préparés pour le système de PN9 contiennent des sous-corpus de différents domaines.

À chaque époque de l'entraînement, un sous-ensemble de corpus est échantillonné de manière stochastique à partir du corpus complet en fonction des règles de distribution qui déterminent la granularité de spécialisation du modèle. L'adaptation du modèle de traduction à un domaine spécifique dépend de la proportion des données du domaine et également de la qualité du corpus sélectionné.

- **Each Iteration is built using weighted sampling over complete data collection**

Corpus Pattern	Weight
IT, MSDN	20
Colloquial	10
Legal	20
Back Translation	10
User Feedback	5
Lexicon	5

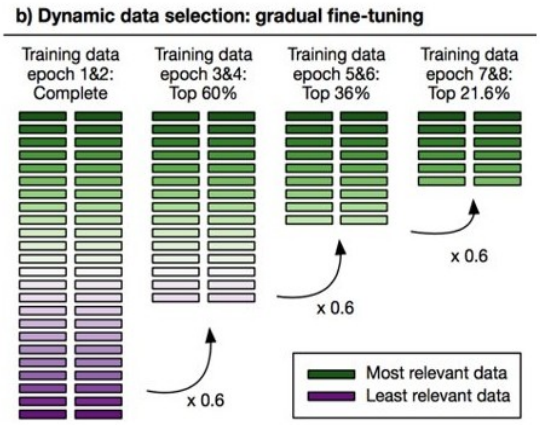

FIGURE 2.3 Échantillonnage dynamique des données

Notes : Figure provient de la documentation interne de Systran

2.2.2 Prétraitement

Chaque ligne doit passer à travers un flux de prétraitement dynamique pendant l'entraînement. Le prétraitement est divisé en deux phases : le prétraitement monolingue et le prétraitement bilingue. Avant de rentrer dans le détail, il est utile de nous familiariser avec quelques ressources linguistiques.

2.2.2.1 Ressources linguistiques

Les ressources linguistiques sont spécifiques à la langue et diffèrent d'une langue à l'autre. Dans le flux de travail de PN9, le prétraitement du corpus est appliqué instantanément. L'entrée de l'entraînement est un corpus brut, pourtant certaines ressources linguistiques doivent être construites à l'avance. Les ressources fondamentales pour l'entraînement comprennent un modèle de tokenisation, un modèle d'alignement par paire de langue ainsi qu'une liste de vocabulaire d'une taille limitée couvrant la paire de langues en jeu. Nous verrons le détail de différentes ressources dans la suite de cette section.

2.2.2.1.1 Modèle de tokenisation

Le modèle de tokenisation est l'une des ressources linguistiques, il permet de segmenter un texte en tokens ou des unités plus petites. Un modèle de traduction traduit un texte tokenisé de langue source à un texte en langue cible.

La tokenisation est la façon de découper un texte en une séquence de tokens (traditionnellement ce sont des unités linguistiques) dont nos vocabulaires font partie. Elle paraît moins complexe dans les langues disposant d'un délimiteur de mots qui est typiquement un espace (telles que l'anglais et français), cependant, toutes les langues ne le font pas (par exemple, des langues asiatiques comme chinois et japonais). Pour une paire de langues donnée (la langue source et la langue cible), nous construisons un modèle de tokenisation par langue vu que les règles de tokenisation changent d'une langue à l'autre.

La tokenisation est très utile pour la SMT, notamment pour traduire un texte dans une langue qui est riche au niveau de la morphologie [***Zalmout et al,. 2017***]. Nous continuons à suivre cette recommandation dans le système de NMT. La fonction de la tokenisation est de convertir les phrases brutes en séquences de tokens, un modèle de tokenisation permet de découper un texte en mots ou même en sous-mots. Dans le fichier de configuration, la tokenisation est une section indépendante de la normalisation.

D'abord nous allons procéder à une normalisation des textes en entrée, il se manifeste par les blocs de prétraitement monolingue (monolingual preprocessing, MPR) et le prétraitement bilingue (bilingual preprocessing, MPR). Cette opération est conçue pour l'identification et la protection de séquences spécifiques telles que les adresses URL (les adresses URL pouvant retrouvées dans un texte obtenu à partir de la page Web, par exemple) à travers une annotation par des caractères établis. Elle permet aussi de normaliser certains caractères (par exemple tous les types de guillemets, des variantes unicode[10]), et même des variantes (telles que des dates) en une représentation. La normalisation applique ses règles indépendamment en source et en cible pendant l'entraînement, et uniquement à la source lors de l'inférence (le processus de traduction du texte de test).

La tokenisation est définie comme un bloc indépendant dans le fichier de configuration. Mais en réalité, la normalisation demande une tokenisation par mots afin d'appliquer ses règles, et une dé-tokenisation après que les règles auraient entrées en vigueur.

Il est nécessaire d'introduire une dé-tokenisation à ce niveau car la diversité des langues ne permet pas de réaliser facilement une tokenisation générique. En conséquence, réappliquer une tokenisation ainsi qu'une sous-tokenisation est indispensable pour que l'application du vocabulaire soit la plus pertinente possible, par exemple, le mode de

10https://en.wikipedia.org/wiki/Variant_form_(Unicode)

tokenisation au niveau de la phrase peut être « agressive » ou d'autres valeurs disponibles[11].

L'étape suivante est un processus à deux niveaux basé sur les textes normalisés. Dans un premier temps, il s'agit d'une tokenisation au niveau de la phrase, qui permet de transformer la phrase normalisée en une séquence de tokens (un token peut être un mot ou une ponctuation) séparés par des espaces. Ensuite une sous-tokenisation est appliquée au niveau du mot, un mot peut être découpé davantage en des fragments de texte (séparés par des espaces). Ce processus est effectué exclusivement après la première tokenisation.

Conformément aux principes du système PN9, nous avons besoin d'entraîner des modèles pour la sous-tokenisation. Nous générons un modèle de NMT à l'aide d'un vocabulaire de taille fixe contenant les mots les plus fréquents (en général de 30000 à 50000 tokens) dans la langue source et la langue cible (les mots dans les deux langues peuvent être fusionnés dans une seule liste ou conservés indépendamment), alors que la traduction est un problème à vocabulaire ouvert. Les mots en dehors du vocabulaire prédéfini (out-of-vocabulary, OOV) font partie des mots inconnus, ils sont généralement annotés par <unk>. Le problème de mots inconnus émerge dans le processus de dé-tokenisation où il s'agit de la génération de traduction. Une phrase en source avec plusieurs mots inconnus provoque une faiblesse en traduction, comme illustré dans la FIGURE 2.4 :

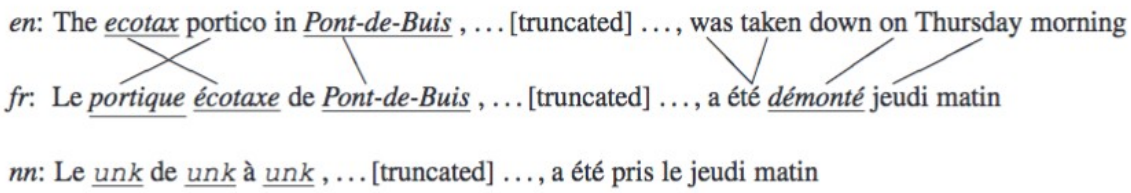

FIGURE 2.4 Exemple du problème de mots inconnus
(*en* : anglais, *fr* : français, *nn* : traduction générée)

Notes : Figure tirée de l'article « Addressing the Rare Word Problem in Neural Machine Translation » ***[Minh-Thang Luong, 2014]***

De ce fait, l'application d'une seule tokenisation par mot augmente le nombre de mots inconnus, surtout pour les langues morphologiquement complexes (par exemple le mot « Autobahnraststätte » en allemand signifie « aire d'autoroute » en français, et se compose de « Auto », « bahn », « rast » et « stätte »). Nous avons besoin d'une tokenisation plus subtile pour capturer le maximum de l'information avec un vocabulaire limité. La sous-tokenisation est une approche effective pour résoudre le problème des OOV [***Sennrich et al,. 2016***], elle consiste à segmenter le texte en unités de sous-mots. L'objectif principal est de faciliter la reproduction de nos expériences sur la NMT avec des unités de sous-mots. Nous présentons deux algorithmes de sous-tokenisation pour nos expérimentations.

11 http://opennmt.net/OpenNMT/options/tokenize/

Byte-pair-encoding (BPE)

L'algorithme Byte pair encoding (BPE) [***Sen et al,. 2016***] permet à un modèle NMT de générer une traduction sur un vocabulaire ouvert par l'encodage des mots inconnus en une séquence d'unités de sous-mots. Cet algorithme repose sur le fait que différentes classes de mots sont traduisibles via des unités plus petites que les mots. Cette hypothèse a été appliquée avec les expériences empiriques de [***Sennrich et al,. 2016***], pour les tâches de traduction de l'anglais vers l'allemand et de l'anglais vers le russe lors de WMT 15, les modèles de sous-tokenisation permettant une amélioration respective jusqu'à 1,1 et 1,3 de point BLEU par rapport à la baseline.

Dans le système PN9, l'entraînement du modèle BPE est réalisé via la boîte à outils du système OpenNMT qui prend un texte tokenisé (par une tokenisation en mots) comme entrée. Tout d'abord, nous représentons chaque token comme une séquence de caractères, ainsi qu'un symbole spécial de fin de mot « ▪ » suivi d'un espace qui nous permet de restaurer la tokenisation d'origine à la fin de l'inférence. Nous comptons itérativement toutes les paires de token et remplaçons chaque occurrence de la paire la plus fréquente (e.g., « études », « étudiant ») par un nouveau symbole (e.g., « étudesétudiant ». ce symbole est temporaire, il est construit pour compter les n-grammes de caractères). Chaque opération de fusion génère un nouveau symbole représentant des n-grammes de caractère. Les n-grammes de caractères fréquents (ou mots entiers) sont finalement fusionnés en un seul symbole qui fera partie du vocabulaire final. (e.g., « étud▪ »)

La sortie du modèle BPE est un texte tokenisé contenant des tokens et des unités de sous-tokens. Les tokens qui sont découpés en sous-tokens contient un délimiteur suivi d'un espace afin de pouvoir être restauré ultérieurement. (e.g., « étud▪ iant▪ »)

SentencePiece

SentencePiece [***Kudo. 2018***] est un système open-source de (sous-)tokenisation et dé-tokenisation conçu pour les systèmes de génération de texte basés sur un réseau de neurones où la taille du vocabulaire est prédéterminée, y compris la NMT. Les outils de sous-tokenisation existant prennent un texte pré-tokenisé en entrée (une première tokenisation au niveau des mots), alors que SentencePiece peut entraîner un modèle de sous-tokenisation directement sur un texte brut, ce qui nous permet de construire un système de NMT purement de bout-en-bout [***Kudo et al,. 2018***]. L'algorithme BPE encode une phrase dans une séquence unique des unités de sous-tokens, alors que l'algorithme de [***Kudo. 2018***] considère qu'une phrase peut être représentée dans plusieurs séquences d'unités de sous-tokens, même avec le même vocabulaire, comme présenté sur la FIGURE 2.5.

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

FIGURE 2.5 Plusieurs séquences de sous-mots encodent la phrase «Hello World»

(Colonne gauche : séquences de sous-mots, colonne droite : identifiant des séquences)

*Notes : Figure étire de l'article «Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates» [**Taku Kudo, 2018**]*

Sentencepiece fait d'abord une tokenisation en mots sur le texte brut (les unités de mots sont séparés par des espaces), ensuite il applique une sous-tokenisation au niveau de l'unité de mot. Il introduit un symbole spécial pour annoter l'endroit où un espace est présenté dans le texte original. Sur une tâche de traduction de l'anglais vers le japonais, il est plus performant que d'autres outils de sous-tokenisation et il est possible d'obtenir un taux de réussite comparable à celui de l'entraînement direct de sous-mots à partir de texte bruts.

Les deux algorithmes sont compatibles dans le système PN9. Nous utilisons BPE pour les langues composées d'alphabets latins, et SentencePiece pour les langues sans délimiteur de mots.

2.2.2.1.2 Modèle d'alignement

[***Simard, 1998***] définit l'alignement comme la relation qui existe entre un texte et sa traduction, une relation qui peut être vue à différents niveaux de granularité. Un système d'alignement multilingue automatique est considéré comme un processus qui prend un corpus multilingue comme entrée et produit une sortie constituée d'appariements mettant en correspondance les segments qui sont en relation de traduction. Il existe plusieurs niveaux d'alignement : alignement de phrases et alignement de mots. Ce dernier consiste à identifier les relations entre les mots (ou des unités de plusieurs mots) d'un textes dans la langue source et ceux du texte dans la langue cible. Celle qui est employée pour la NMT est un alignement de mots.

L'alignement a lieu dans le processus de prétraitement bilingue. Un modèle d'alignement prend une paire de phrase dans le corpus d'entraînement en entrée et aligne chaque token dans la langue source et dans la langue cible par la création des index de mot. Cette opération vise deux objectifs : constituer une référence pour filtrer des paires de

phrases parallèles avec des mots relativement non alignés, et pour injecter des annotations lorsque des entités ne doivent pas être tokenisées ni traduites (e.g., adresses URL).

Nous employons un outil d'alignement fast_align pour l'entraînement du modèle d'alignement. Fast_align est un aligneur de mots simple, rapide et non supervisé, les entrées pour fast_align doivent être tokenisées (par mot) et alignées en phrases parallèles. Chaque ligne est une phrase de la langue source et sa traduction dans la langue cible. Voici un exemple de l'entrée avec des paires de phrases extraites du corpus parallèle français-chinois et leur redirection dans un nouveau texte avec une phrase dans la langue source et cible dans une même ligne, séparées par un délimiteur.

```
1 电 池 警 报 : ||| Alerte de batterie :
2 仅 CLI 管 理 软 件 ||| Logiciel de gestion de type CLI uniquement
3 SNMP 陷 阱 MIB ||| MIB de déroutement SNMP
4 2 0 0 8 年 9 月 2 2 日 ||| 2 2 septembre 2 0 0 8
5 发 行 版 6 . 1 . 2 ||| Version 6 . 1 . 2
```

FIGURE 2.6 Exemple de l'entrée (chinois-français) du modèle d'alignement

Une fois le modèle d'alignement entraîné, nous pouvons l'utiliser pour aligner des paires de mots. Chaque mot dans la phrase source et cible possède un index compris entre 0 et le nombre de mots moins un. La Figure 2.7 présente un exemple de sortie d'alignement pour quelques paires de phrases anglais-français.

```
INFO loading alignment model from file '/root/corpus/en_fr/word_align/enfr/forward.probs

INFO SRC: It's sometimes necessary not to tell the truth.
INFO TGT: Il est parfois nécessaire de ne pas dire la vérité.
INFO ALIGN ID:  0-0 1-1 2-1 3-2 4-3 5-5 5-6 6-4 7-7 8-8 9-9 10-10
INFO SRC: This type of thing never used to happen here.
INFO TGT: Ce genre de chose ne se passait jamais ici d'habitude.
INFO ALIGN ID:  0-0 1-1 2-2 3-3 4-4 4-7 5-11 6-9 6-10 7-5 7-6 8-8 9-12
INFO SRC: I work to live, but I don't live to work.
INFO TGT: Je travaille pour vivre, mais je ne vis pas pour travailler.
INFO ALIGN ID:  0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-7 9-9 10-8 11-10 12-11 13-12
INFO SRC: He likes sarcasm a lot.
INFO TGT: Il apprécie beaucoup le sarcasme.
INFO ALIGN ID:  0-0 1-1 2-3 2-4 3-2 4-2 5-5
INFO SRC: Whatever shall I do?
INFO TGT: Qu'est-ce que je devrais faire?
INFO ALIGN ID:  0-0 1-1 1-2 2-5 2-6 2-7 3-8 4-9
```

FIGURE 2.7 Exemple de la sortie du modèle d'alignement

2.2.2.1.3 Vocabulaire

Cette ressource linguistique a été représentée dans la section 2.1.2.1.1. Le vocabulaire est une liste de tokens de taille fixe, les tokens en dehors de cette liste ou ceux qui sont peu fréquents sont des mots inconnus. Nous envisageons de générer une liste de vocabulaires par langue. La génération de vocabulaire est juste après la phase tokenisation et par le système OpenNMT.

2.2.2.2 Prétraitement monolingue

Les ressources linguistiques décrites précédemment sont préparées avant l'entraînement du modèle et fusionnées lors du processus de prétraitement. Dans un premier temps, un prétraitement monolingue est réalisé pendant cette phase.

Le prétraitement monolingue applique des règles indépendamment sur la source et la cible pendant l'entraînement. Il s'agit de normaliser des caractères ou des nombres. Par exemple, nous pouvons normaliser les caractères du chinois traditionnel en chinois simplifié par l'application des règles de «*localization_rules*» (voir fig. 2.8).

```
"mpreprocess": {
    "source": {
        "normalization": {
            "character": {
                "icu_rules":["${CORPUS_DIR}/xx/quote_icu.rules"],
                "numbers": "none",
                "punctuation": "none",
                "ligatures": true,
                "tatweel": false,
                "tashkil": false,
                "alif_maqsura": false,
                "ta_marbuta": false,
                "hamza": false,
                "maddah": false,
                "nunation": false
            }
        }
    },
    "target": {
        "normalization": {
            "character": {
                "numbers": "none",
                "punctuation": "none",
                "ligatures": true,
                "tatweel": false,
                "tashkil": false,
                "alif_maqsura": false,
                "ta_marbuta": false,
                "hamza": false,
                "maddah": false,
                "nunation": false
            }
        },
        "localization_rules":["${CORPUS_DIR}/zh/tc2sc.rules"]
    }
```

FIGURE 2.8 Extrait de la configuration pour le prétraitement monolingue

2.2.2.3 Prétraitement bilingue

Dans cette étape, nous procédons une normalisation sur les phrases parallèles issues du corpus d'entraînement, telle que identifier et protéger des paires d'entités, filtrer des paires de phrases de longueur non équilibrée et filtrer celles qui avec des mots relativement non alignés par comparaison de l'index d'annotation dans la phrase source et son mapping dans la phrase cible.

Les identités à protéger consistent en des adresses URL, des formatages tels que des balises dans le code source HTML. L'entrée acceptable par le système de traduction est un texte brut alors que la traduction demandée peut concerner un document formaté, parfois nous avons besoin de traduire une page web en conservant les balises HTML. À cet effet, nous tendons à injecter une annotation au tour des séquences à protéger. Nous utilisons différents marqueurs en fonction de la nature des séquence à protéger, par exemple <unk> est une marqueur d'annotation spécifique pour les OVVs.

Nous voulons éliminer les paires de phrases contenant de nombreux mots non alignés en fonction de la valeur de perplexité générée par le modèle d'alignement. Plus la valeur (absolue) de perplexité pour une paire de phrase est petite, meilleur sera l'alignement.

En effet, comme expliqué dans la section 2.2.2.1.1, le prétraitement bilingue ne s'applique pas tant que le texte n'est pas tokenisé. Par exemple, filtrage des paires de langue non alignées et injection des marqueurs d'annotation nécessitent une tokenisation basique (par mots). À la fin de ce stade, nous restaurons le texte tokenisé en faisant une dé-tokenisation.

2.2.3 Entraînement et post-traitement

Une fois les ressources linguistiques disponibles et les données du corpus d'entraînement prétraitées, nous pouvons entraîner les modèles de NMT. Le système PN9 repose sur l'architecture Transformer qui accélère le processus d'entraînement du modèle d'un ordre de grandeur, le Transformer peut accélérer le processus d'entraînement de quelques jours à quelques heures. L'amélioration de performance de cette nouvelle système nous permet de réaliser un entraînement infini dans la condition de GPU[12] disponibles.

Une époque est une cycle d'apprentissage en passant toutes les données de corpus d'entraînement. Notre système permet de générer un modèle à la fin de chaque époque, nous évaluons le modèle généré et éventuellement continuons l'entraînement à partir de ce dernier. Comme le corpus d'entraînement est construit à travers un échantillonnage stochastique, nous ne retrouvons presque jamais le même corpus d'entraînement à chaque fois de l'entraînement, ce qui aboutit inévitablement à une reprise du prétraitement.

12Processeur graphique https://fr.wikipedia.org/wiki/Processeur_graphique

Une époque d'entraînement dure environ 3h, mais cela dépend de la capacité et de la disponibilité de GPU et de la taille du corpus d'entraînement. Une fois l'entraînement terminé, le processus de traduction se déclenche automatiquement sur un corpus de test.

Le décodeur est essentiellement un modèle linguistique qui joue un rôle de classifieur multi-classes dans le système de NMT, il génère la traduction mot par mot pour prédire le mot suivant en traduction en calculant la probabilité que chaque token du vocabulaire apparaisse dans la condition actuelle (la représentation vectorielle de la phrase en source générée par l'encodeur et le mot actuel dans la langue source). Si un mot en source n'est pas présenté dans le vocabulaire ou il acquiert une faible probabilité (mot rare), il sera annoté par <unk> dans la traduction.

La traduction générée par le décodeur a besoin d'une dé-tokenisation. Un modèle de dé-tokenisation est intégré dans le système PN9 dans l'objectif d'éliminer les espaces superflus que nous avons ajoutés lors de la tokenisation.

Le post-traitement est appliqué à la traduction générée par le décodeur après l'inférence. Il transfère des formatages et quelques informations originales à partir de la phrase source à la traduction en cible. Par exemple, le post-traitement restaure les valeurs protégées (adresses URL, formatages HTML et les <unk>) dans le processus de dé-normalisation.

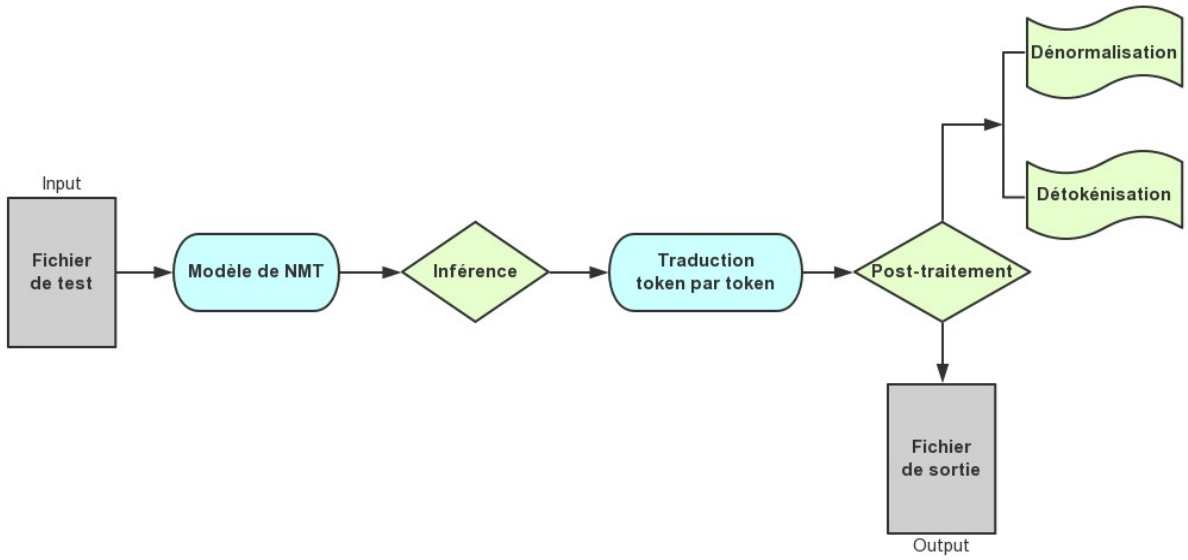

FIGURE 2.9 Flux de travail de l'inférence

2.2.4 Méthode d'évaluation

Nous disposons un catalogue de modèles pour visualiser des scores BLEU. BLEU est une mesure couramment utilisée pour évaluer les performances de la traduction. Nous utilisons un script qui prend la prédiction de traduction et la référence dans la langue cible en entrée et il produit systématiquement un score BLEU par document.

« plus une traduction automatique est proche d'une traduction humaine professionnelle, mieux c'est » - c'est l'idée centrale derrière BLEU. BLEU a été l'une des premières métriques à revendiquer une corrélation élevée avec les jugements humains de qualité, et reste l'une des métriques automatisées les plus populaires et les moins coûteuses. [***ref. Wikipédia***[13]]

2.3 Modèle générique et spécialisé

La spécialisation est le point clé de notre expérience. Nous effectuons un processus d'entraînement de fine-tuning, qui permet d'adapter au fur et à mesure un modèle générique à un modèle spécialisé. Le processus de spécialisation est schématisé dans la FIGURE 2.10.

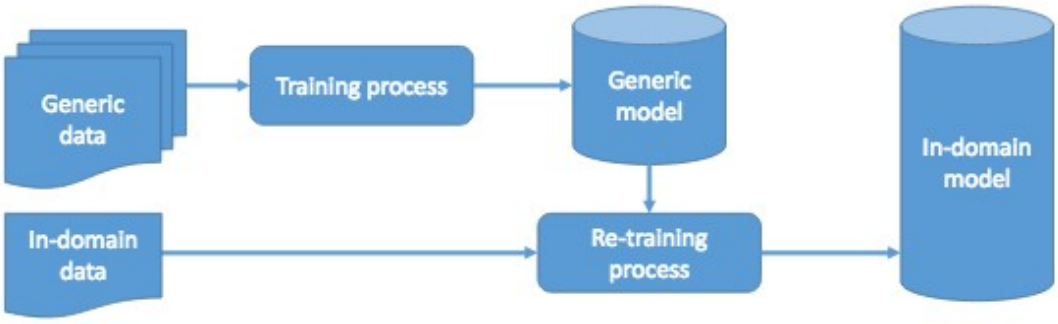

FIGURE 2.10 Flux de travail de l'adaptation du modèle générique aux domaines spécialisés

Notes : Figure issue de l'article « Domain specialization: a post-training domain adaptation for Neural Machine Translation »
[Servan et alt,. 2016]

En premier lieu, nous entraînons des modèles de traduction génériques par l'échantillonnage des données multi-domaines. En second lieu, nous choisissons le meilleur modèle générique par l'évaluation de ces modèles (traduction du corpus de test avec le modèle, puis comparaison des scores BLEU et analyse de la traduction prédite). Ensuite, nous mettons en application le processus de DA via la spécialisation. Nous continuons

13 https://fr.wikipedia.org/wiki/BLEU_(algorithme)

d'entraîner le modèle en quelques itérations supplémentaires à partir du meilleur modèle générique en transmettant seulement des données du domaine pendant l'échantillonnage.

La spécialisation permet d'adapter progressivement un modèle générique à un domaine spécifique, et cela évite de recommencer le processus d'entraînement.

2.4 Classification

Après avoir exploré les processus de l'entraînement de modèles de traduction dans la partie précédente, dans cette partie nous présentons les méthodes de classification utilisées et l'objectif d'utilisation.

La classification dans le domaine d'apprentissage automatique consiste à une classe ou catégorie à chaque objet (ou individu) à classer, en se basant sur des données statistiques [***ref. Wikipédia***[14]]. Nous introduisons les méthodes de classification supervisées dans ce travail de recherche afin de prédire les domaines de textes dans le corpus de test, afin de choisir un modèle de traduction du même domaine de textes à traduire, et ainsi obtenir une bonne traduction en langue cible.

Le flux de travail pour la tâche de classification est divisé en 2 parties. D'abord il s'agit éventuellement d'un pré-traitement sur le document d'entrée, qui consiste à éliminer les mots grammaticaux, transformer les lettres majuscules en minuscules, normaliser les accents, tokeniser les textes autour de l'espace. Ensuite, nous choisissons deux méthodes de classification : entraînement d'un modèle de langue (language model, LM) basé sur les n-grammes de mots et utilisation de l'algorithme Naïve Bayes.

Pour entraîner un LM, nous nous servons de la boîte à outil KenLM. KenLM est un modèle probabiliste qui permet d'estimer, filtrer et interroger un LM [***Kenneth Heafield, 2011***]. LM est basé sur l'algorithme de n-gramme de mots. Pour une phrase S ($S = W_1, W_2, \ldots W_m$), sa probabilité peut être présentée comme suit :

$$p(w_1, w_2, \ldots, w_m) = p(w_1) * p(w_2|w_1) * p(w_3|w_1, w_2) \ldots . p(w_m|w_1, .., w_{m-1})$$

FIGURE 2.11 Algorithme de N-gramme

Les méthodes Naïve Bayes constituent un ensemble d'algorithmes d'apprentissage supervisés et probabilistes reposant sur l'application du théorème de Bayes. Il est naïf du fait de son hypothèse d'indépendance conditionnelle entre des caractéristiques (features). Nous utilisons l'algorithme multinomial, parce qu'il est adapté à la classification avec des features discrètes (par exemple, le nombre de mots pour la classification du texte).

14https://fr.wikipedia.org/wiki/Classement_automatique

Dans notre expérience, les features d'entrée pour ce classifieur est une représentation de TF-IDF. Le TF-IDF est une mesure statistique permettant d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le TF (term frequency) est le nombre d'occurrences du mot dans le document, et l'IDF (Inverse Document Frequency) mesure le nombre de documents dans le corpus étudié qui contient le mot donné, rapporté à l'ensemble des documents analysés.

Il estime la classe d'un texte par calculer la probabilité conditionnelle que le texte appartient à une classe sachant les features de ce texte. Pour un texte à classifier, nous obtenons d'abord sa représentation vectorielle soit **x**=(F_1, F_2, … , F_n) dont *n* est le nombre de features extraits (Les features sont de variables indépendantes), et ainsi, la probabilité que le texte est de la classe C peut être présentée par le formulaire suivant :

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\,p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}$$

FIGURE 2.12 Algorithme Bayésien Naïf
fondé sur le théorème de Bayes

Les deux classifieurs traitent un texte comme un « sac de mots », et assignent la classe correspondant à la plus grande probabilité parmi toutes les classent dans le corpus d'entraînement à un texte du corpus de test. En dépit de leur simplicité, ils fonctionnent bien pour les tâches de classification de documents.

Traditionnellement, nous voulons prédire le domaine d'un document dans le corpus de test et utiliser le modèle de traduction du domaine pour produire la traduction. Dans ce travail de recherche, nous cherchons à comparer la méthode traditionnelle et notre approche consistant à calculer, pour chaque phrase du document, la probabilité que la phrase appartient à chaque classe.

Notre approche repose sur l'hypothèse que, pour une phrase d'un document du domaine spécialisé donné, elle peut ne pas appartenir au même domaine du document lui-même. Par exemple, une phrase en français « Des compagnies qui bloquent l'accès à la messagerie instantanée ou à Facebook. » est extraite d'un document du domaine du discours oral, pourtant il est difficile d'y accorder une classe de discours oral. Ainsi, nous explorons la classe la plus probable de chaque phrase du document pour avoir des informations plus précises.

2.5 Pondération

Un système de traduction automatique renvoie généralement les meilleurs candidats de traduction, cependant, nous avons besoin d'un critère pour classer ces candidats en

fonction des taux de confiance de plusieurs granularités par rapport à la phrase source. Il s'agit d'un reranking de n-best liste (n-best list reranking), le score de confiance est le résultat de la mise à l'échelle (scalling) de la perplexité qu'un candidat est la traduction de la phrase source. Plus la valeur absolue du score de confiance est petite, mieux c'est.

Généralement, la traduction prédite par le système de MT comprend plus d'un candidat, et un score de confiance est un seuil qui détermine le score correspondant le plus bas acceptable dans la traduction. Pour chaque phrase, nous allons choisir un seul candidat de traduction assigné par le meilleur score de confiance, et pondérer ce score respectivement par la probabilité que cette phrase appartient à chaque classe, ensuite contraster les scores pondérés correspondant à chaque classe. Enfin la classe disposant le meilleur score pondéré sera la classe prédite pour la phrase. Ainsi, nous savons choisir le modèle de traduction pour traduire la phrase. De cette manière nous formons une traduction « synthétique » en sortie, et nous calculons également son score BLEU.

Le processus de traduction est exécuté automatiquement après chaque entraînement du modèle, ce qui nous permet de disposer de toutes les traductions prédites par chaque modèle (génétique et spécialisé). Ceci nous donne accès à toutes les traductions prédites, après avoir prédit la classe d'une phrase, nous pouvons facilement analyser la traduction correspondante.

3. Expérimentations

Sommaires

3.1 Introduction

Dans ce chapitre, nous décrivons nos corpus utilisés ainsi que les prétraitements effectués, les deux sous-tâches de notre expérimentation, et des travaux de suivi que nous avons mis en œuvre. D'abord dans la section 3.2, nous abordons la première tâche consistant à générer des modèles de NMT, y compris l'évaluation de notre méthode en comparaison d'une baseline. Ensuite, nous présentons dans la partie 3.3 la tâche de classification ainsi que les prétraitements sur le corpus d'entraînement. À la fin de cette phase, nous évaluons les classifieurs entraînés. Ensuite la section 3.4 consiste à générer des traductions pour les documents du corpus de test en adaptant un classifier. Enfin nous terminons ce chapitre par évaluer les traductions obtenues en comparant les scores BLEU.

3.2 Partie I : Entraînement des modèles de traduction

3.2.1 Préparation des données propres

Dans la base de données de Systran, nous trouvons un corpus parallèle par paire de langues constitué de documents parallèles classés en fonction du domaine. Ils sont généralement issus du web et des corpus open source permettant la création du corpus à la fois multilingues et multi-domaines. L'acquisition de corpus pour l'entraînement des modèles de traduction se fait en plusieurs étapes. Premièrement, le téléchargement de sites puis l'extraction du texte. Ensuite, le nettoyage ainsi que la segmentation des données, pour enfin pouvoir aligner les segments (dans le cas des données multilingues). Une chaîne de traitement est présentée dans la FIGURE 3.1 :

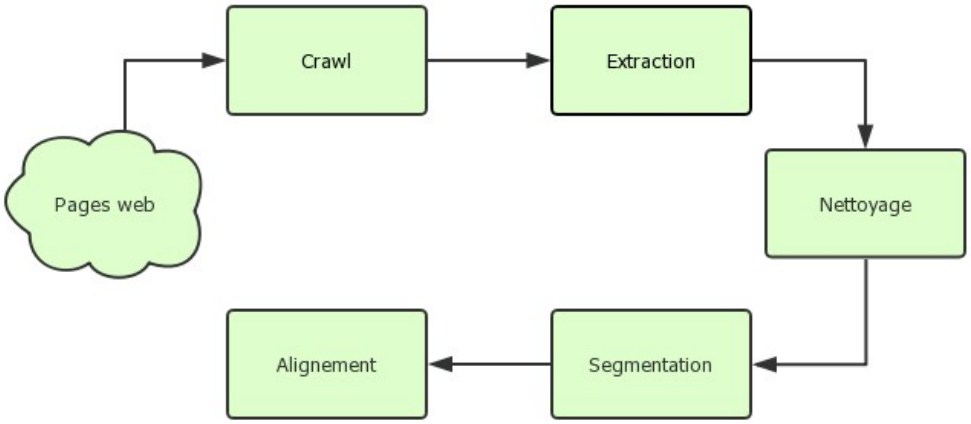

FIGURE 3.1 Flux de travail pour la création de corpus

D'abord la sélection et analyse des sites sont nécessaires afin de définir s'ils contiennent de données parallèles en langues voulues, ensuite il faut « crawler »[15] le site. Une fois le crawl terminé, on cherche à apparier les documents afin de trouver pour chaque fichier en langue source son fichier correspondant en langue cible. L'étape suivante consiste à supprimer dans les textes bruts les lignes vides et réduire les espaces multiples, supprimer les émoticônes, les caractères corrompus et les entités HTML, etc. Lors de la phase d'alignement, Systran utilise l'outil hunAlign[16] qui permet d'aligner du texte parallèle, entre une phrase du texte source et une phrase du texte cible. Cet outil prend en entrée un texte segmenté (une phrase par ligne pour la source et la cible) et produit en sortie un bitexte[17]. Enfin, nous construisons un corpus composé des paires de documents parallèles nous permettant de bénéficier, pour chaque paire, du document en langue source avec une phrase par ligne, et sa traduction correspondant au même index de ligne dans le document en langue cible.

15Le «crawl» est le téléchargement des pages web sans passer par un navigateur.
16https://github.com/danielvarga/hunalign
17Un bitexte est le résultat de l'alignement d'un texte et de sa traduction dans un document fusioné

Nous envisageons d'entraîner un modèle de traduction qui permet de traduire un texte français en chinois. L'objectif de la tâche est d'étudier et de comparer différentes méthodes afin d'obtenir la meilleure traduction. Le corpus français-chinois se compose de documents provenant de divers domaines. Nos données incluent plus de 23 millions (23 738 695) de paires de phrases stockées dans 32 paires de documents (une paire consiste en un document français et un document chinois en parallèle). Les données sont issues du corpus OpenSubtitles[18], TedTalks[19], Tatoeba[20], UN[21], TAUS[22], ainsi que des données synthétiques[23] et celles qui sont « crawlées » à partir des pages web, prétraitées et restructurées par l'équipe d'ingénierie de traitement du langage naturel. L'échantillonnage de données pour le corpus d'entraînement diffère en fonction de tâche de traduction, le détail se trouve dans les sections 3.2.2 et 3.2.3.

En ce qui concerne le corpus de test, nous choisissons 13 paires de documents de différents domaines du corpus total (le cas du corpus français-chinois), et extrayons 500 lignes aléatoires par documents pour construire notre corpus de test tout en réservant le même nom du document d'origine, comme présenté dans le Tableau 3.1. Les phrases dans le corpus du test ne doivent pas apparaître dans le corpus d'entraînement.

Unité : phrases / langue

Nom du document	Taille
IT	2500
Legal	1500
OpenSubtitles	500
ted-talks	500
Manual	500
News	500
Patent	500
TOTAL	6500

TABLE 3.1 Corpus de test

(Pour les documents dont les noms contiennent le motif inscrit dans la première colonne, leur nombre total de lignes est indiqué dans la deuxième colonne)

Nous avons donc 6500 paires de phrases parallèles dans le corpus de test, les données d'entraînement seront extraites de 23 732 195 paires de phrases restantes.

18Une collection de sous-titres de films traduits de http://www.opensubtitles.org/.
19Un corpus contient des exposés de TED en anglais et des traductions en plusieurs langues.
20Un site web alimenté de manière collaborative, proposant une base de données de phrases traduites dans toutes les langues,
21Une collection de documents traduits des Nations Unies
22Données des mémoires de traduction des entreprises commerciales. Beaucoup du domaine manuel du logiciel / ordinateur.
23Les données synthétiques sont des données (des mots,locution ou phrases) construites artificiellement plutôt que générées par des événements réels.

Revenons sur le corpus de test, il s'agit de 2500 paires de phrases pour le domaine d'informatique (IT), 1500 pour celui de juridique (Legal), 1000 pour celui de discours oral (OpenSubtitles, ted-talks), etc.

Nous suivons le processus du prétraitement présenté dans le chapitre précédent, tout en respectant la particularité de la langue à traiter. Le MPR du texte français consiste à normaliser les guillemets (et éventuellement d'autres transformations au niveau des caractères) alors qu'il s'agit de transformer les caractères chinois traditionnels à simplifiés. Ensuite, nous travaillons au niveau des fichiers parallèles. Il s'agit de charger le modèle d'alignement, filtrer les caractères bruités (les caractères cyrilliques dans un document français par exemple).

Le processus de tokenisation est un traitement monolingue, nous choisissons le mode de tokenisation en fonction de langue à traiter. Le français est une langue composée de l'alphabet latin disposant d'un délimiteur de mots (un espace dans ce cas-là), nous appliquons d'abord une tokenisation en mots, puis une sous-tokenisation par le BPE intégrée dans le système sur la sortie de cette dernière étape. À la sortie de chaque étape, nous avons un texte tokenisé à différents niveaux comme présenté ci-dessous (Le caractère spécial « ▪ » que l'on appelle un « joiner » est utilisé pour annoter l'endroit où il s'agit d'une dé-tokenisation à la fin de l'inférence) :

Texte d'entrée	Car les mères qui allaitaient, ou qui avaient des enfant en bas âge achetaient quelque chose qu'elles croyaient être sain, mais qui était en fait très toxique.
Tokenisation au niveau de mots	Car les mères qui allaitaient, ou qui avaient des enfant en bas âge achetaient quelque chose qu'elles croyaient être sain , mais qui était en fait très toxique .
Tokenisation par BPE	Car les mères qui alla▪ it▪ aient ▪, ou qui avaient des enfant en bas âge ache▪ taient quelque chose qu ▪'▪ elles croy▪ aient être sain ▪, mais qui était en fait très toxique ▪.

TABLE 3.2 Exemple d'un texte français tokenisé

(Extrait du corpus TED talks)

Le système SentencePiece a montré sa performance et sa robustesse sur le problème de tokenisation des langues ne disposant pas de délimiteur de mots (telles que le japonais et le chinois). Nous réalisons la (sous-)tokenisation du texte chinois par l'application du SentencePiece qui prend un texte brut comme entrée et produit un texte tokenisé. À la différence du BPE, il utilise un caractère spécial «▁» (U+2581) afin d'indiquer qu'un espace doit être introduit pendant la dé-tokenisation. L'entrée et la sortie du SentencePiece sont comme suit :

Texte d'entrée	[COMPANY] 推荐使用 Windows Vista®Business 商用版操作系统。
Tokenisation par SentencePiece	▁[COM P AN Y] ▁推荐 使用 ▁Windows ▁Vista ® Business ▁商用 版 操作系统 。

TABLE 3.3 Exemple d'un texte chinois tokenisé

(Extrait du corpus TAUS)

3.2.2 Entraînement des modèles génériques

Notre baseline concerne les modèles génériques. Après une évaluation approximative de la qualité des 23 732 195 paires de phrases préparées pour l'entraînement, nous avons élaboré un premier plan de distribution des données et l'échantillonnage :

Unité : phrases / langue

Nom du document	Taille	Extraction	Proportion
Legal	15284268	2000000	0.6472
OpenSubtitles	1905540	70425	0.0228
tatoeba	15703	31000*	0.0100
ted-talks	138488	270000*	0.0874
IT	4526267	70425	0.0228
Manual	165252	70425	0.0228
Patent	278639	70425	0.0228
News	59698	119000*	0.0385
MISC	241	300*	0.0001
Finance	40210	79000*	0.0256
Generic	486290	300000	0.0971
Names	831599	9000	0.0029
TOTAL	23732195	3090000	1.0000

TABLE 3.4 Corpus parallèle de français-chinois et l'échantillonnage

(Pour les documents dont les noms contiennent le motif inscrit dans la première colonne, leur nombre total de lignes est indiqué dans la deuxième colonne et le nombre de lignes à extraire est inscrit dans la troisième colonne. La dernière colonne présente la proportion de l'extraction par rapport au total de l'extraction. Notes : Il s'agit de certaines phrases en double pour des chiffres avec « * »)

Plus de la moitié des données du corpus proviennent des documents relevant du domaine juridique. D'un côté, les données du domaine juridique proviennent du corpus des Nations Unies (UN) qui sont plus génériques et fiables ; de l'autre côté, nous sommes confrontés au défi du déséquilibre des données. À part des données du domaine juridique, nous avons des données provenant du domaine de l'informatique (IT et Manual, ce dernier

est assez proche du domaine IT), de discours oral (tatoeba, ted-talks, MISC), etc. Tandis que des données du domaine de la presse (News) et de la finance (Finance) sont aussi importantes pour l'entraînement du modèle et nous n'en disposons pas assez.

Les données extraites constituent notre corpus d'entraînement, qui passe d'abord par la phase du prétraitement, puis nous commençons un premier entraînement pendant 30 époques (chaque époque d'entraînement prend environ 2h). Ainsi, nous disposons d'un modèle générique par époque, l'évaluation de ces modèles se fait à travers le corpus de test, nous précisons l'évaluation dans le chapitre suivant.

3.2.3 Entraînement des modèles spécialisés

L'entraînement de modèles spécialisé par « fine-tuning » s'appuie sur les modèles génériques. Nous évaluons ces modèles en examinant les performances de traduction sur le corpus de test.

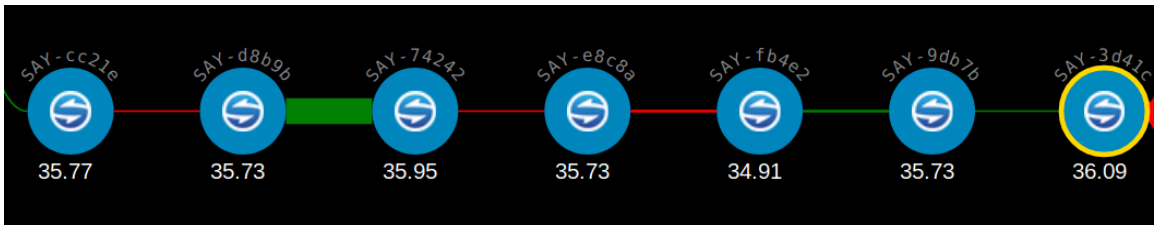

FIGURE 3.2 Les scores BLEU calculés à partir de la traduction générée par les 5 derniers modèles génériques (époque 26-30)

La Figure 3.2 est un extrait du schéma de la présentation des scores BLEU. Chaque nœud représente le score BLEU moyen de la traduction de tous les fichiers du corpus de test générée par le modèle courant. Dans une perspective macro, le modèle générique à la 30ème époque est plus performant.

Dans un point de vue micro et afin de maximiser la performance des modèles spécialisés, nous étudions la micro-moyenne du score BLEU pour les fichiers de test du domaine, et choisissons un modèle générique par domaine en tant que nouvelle baseline de ce domaine. Le détail de l'évaluation des modèles générique sera précisé dans la section 3.2.4.

Nous nous appuyons sur les modèles de baseline pour réaliser la spécialisation. Le Tableau 3.5 ci-après présente notre deuxième plan de distribution de données d'entraînement.

Unité : phrases / langue

Domaine	Nom du document	Taille	Extraction	Total
IT	IT	4526267	2935500	3090000
	Manual	165252	154500	
Discours oral	OpenSubtitles	1905540	1905540	2059972
	tatoeba	15703	15703	
	ted-talks	138488	138488	
	MISC	241	241	
Legal	Legal	15284268	3090000	3090000

TABLE 3.5 Corpus parallèle de français-chinois et l'échantillonnage pour la spécialisation

(La première colonne présente la catégorie des documents. Pour les documents dont les noms contiennent le motif inscrit dans la deuxième colonne, leur nombre total de lignes est indiqué dans la troisième colonne et le nombre de lignes à extraire est inscrit dans la quatrième colonne.)

Parmi les corpus parallèles français-chinois existant dans notre base de données, nous avons finalement choisi ces 3 domaines ayant une multitude de données (la quantité de données par domaine peut être légèrement différente). Avec ce plan d'échantillonnage, nous entraînons 3 (groupes de) modèles spécialisés consistant en modèles d'informatique, modèles sert à modéliser des discours, et modèles du juridique) en 5 itérations de plus à partir du modèle de baseline le plus performant de chaque domaine, suivant les mêmes échelles de traitements utilisées pour la génération de modèles génériques.

3.2.4 Évaluation

Nous avons entraîné des modèles de NMT en réalisant une spécialisation de fine-tuning, il s'agissait d'un entraînement de 30 époques pour les modèles génériques, et 5 itérations de plus pour chaque modèle spécialisé à partir du modèle générique le plus performant sur les fichiers de test du domaine.

L'évaluation de ces modèles est réalisée par la mesure d'évaluation BLEU. Les 3 tableaux suivants incluent les scores BLEU obtenus pour tous les modèles entraînés que ce soient génériques ou spécialisés. Nous donnons quelques explications afin de comprendre les informations sur ces tableaux :

La moitié supérieure du tableau concerne les modèles génériques, et l'autre moitié porte sur les modèles du spécialisés à l'informatique / discours oral / juridique. La colonne *Model ID* indique le modèle avec le nombre d'itérations en index, les champs *[test file]* consiste à un fichier de test, la colonne *Moyenne* affiche la micro-moyenne de scores BLEU concernant les fichiers du domaine par modèle, la ligne *Moyenne* présente la micro-moyenne des scores BLEU pour chaque fichier de test ; la couleur verte indique les meilleurs scores pour chaque fichier du test, la couleur jaune présente la meilleure micro-moyenne de score BLEU pour les modèles génériques / spécialisés. Nous utilisons le terme

« micro-moyenne » car chaque tableau ne présente que les informations sur les fichiers du domaine.

Modèles Generiques						
Model ID	[test file]	[test file]	[test file]	[test file]	[test file]	**Moyenne**
StillOlaf 01	19.71	25.04	17.31	18.44	19.06	19.91
StillOlaf 02	26.78	35.7	25.14	23.87	23.64	27.03
StillOlaf 03	30.64	40.24	29.41	27.47	26.31	30.81
StillOlaf 04	34.16	44.53	32.47	30.14	29.87	34.23
StillOlaf 05	34	46.58	33.79	30.51	30.17	35.01
StillOlaf 06	35.69	46.94	34.22	31.7	30.41	35.79
StillOlaf 07	34.84	45.89	33.8	31.6	31.2	35.47
StillOlaf 08	35.87	48.55	36.56	33.12	32.6	37.34
StillOlaf 09	35.81	48.31	36.07	32.57	31.82	36.92
StillOlaf 10	36.6	47.08	36.45	33.5	30.79	36.88
StillOlaf 11	36.92	48.72	37.33	33.91	33.03	37.98
StillOlaf 12	36.64	48.52	36.49	32.7	32.53	37.38
StillOlaf 13	36.46	43.84	35.44	30.44	29.85	35.21
StillOlaf 14	37.05	48.09	36.46	32.58	31.59	37.15
StillOlaf 15	38.15	50.24	38.26	34.42	33.22	38.86
StillOlaf 16	37.36	48.89	37.45	33.55	32.81	38.01
StillOlaf 17	35.78	45.99	35.34	31.14	31.22	35.89
StillOlaf 18	38.27	51.45	38.29	34.82	33.79	39.32
StillOlaf 19	37.79	50.01	38.11	34.39	34.01	38.86
StillOlaf 20	38.17	49.93	38.27	35.15	33.22	38.95
StillOlaf 21	38.22	50.51	38.65	34.42	33.51	39.06
StillOlaf 22	37.93	48.8	37.46	33.26	31.57	37.80
StillOlaf 23	38.46	51.3	37.64	34.39	33.54	39.07
StillOlaf 24	38.57	51.36	38.98	34.85	33.97	39.55
StillOlaf 25	38.5	51.65		35.14	33.9	39.80
StillOlaf 26	38.64	51.42	39.14	35.65	34.16	39.80
StillOlaf 27	38.57	51.03	38.46	35.86	34.36	39.66
StillOlaf 28	38.3	49.77	37.97	34.44	32.83	38.66
StillOlaf 29	38.63	51.17	38.65	35.64	34.02	39.62
StillOlaf 30	38.65	51.39	39	35.98	34.02	39.81
Moyenne	36.039	47.431	35.607	32.522	31.567	36.66

Modèles Spécialisés						
Model ID	[test file]	[test file]	[test file]	[test file]	[test file]	**Moyenne**
StillOlaf_31	43.6	58.22	45.39	41.77	38.6	45.52
StillOlaf_32	44.76	59.29	47.59	42.62	39.32	46.72
StillOlaf_33	45.7	60.44	48.4	43.27	39.93	47.55
StillOlaf_34	45.9	60.49	49	43.57	40.19	47.83
StillOlaf_35	47.01	61.42	49.08	44.58	41.7	48.76
Moyenne	45.39	59.97	47.89	43.16	39.95	47.27

TABLE 3.6 Scores BLEU pour les modèles génériques & modèles spécialisés (informatique) pour les fichiers de test du domaine d'informatique

Modèles Generiques			
Model ID	[test file]	[test file]	**Moyenne**
StillOlaf 01	9.56	5.13	7.35
StillOlaf 02	13.49	6.67	10.08
StillOlaf 03	16.07	7.67	11.87
StillOlaf 04	17.72	10.37	14.05
StillOlaf 05	17.71	11.34	14.53
StillOlaf 06	18.86	10.69	14.78
StillOlaf 07	17.8	7.49	12.65
StillOlaf 08	17.78	11.73	14.76
StillOlaf 09	18.89	10.89	14.89
StillOlaf 10	19.26	10.42	14.84
StillOlaf 11	19.28	11.41	15.35
StillOlaf 12	19	11.09	15.05
StillOlaf 13	16.97	4.45	10.71
StillOlaf 14	17.79	10.41	14.10
StillOlaf 15	18.98	10.64	14.81
StillOlaf 16	18.65	11.07	14.86
StillOlaf 17	17.77	5.32	11.55
StillOlaf 18	18.42	12.39	15.41
StillOlaf 19	18.46	11.71	15.09
StillOlaf 20	19.09	11.8	15.45
StillOlaf 21	18.12	11.52	14.82
StillOlaf 22	18.54	8.95	13.75
StillOlaf 23	18.97	12.12	15.55
StillOlaf 24	18.61	11.97	15.29
StillOlaf 25	18.41	11.64	15.03
StillOlaf 26	18.34	12.15	15.25
StillOlaf 27	18.89	11.78	15.34
StillOlaf 28	18.44	9.86	14.15
StillOlaf 29	17.86	12.09	14.98
StillOlaf 30	18.72	11.3	15.01
Moyenne	17.88	10.20	14.04

Modèles Spécialisés			
Model ID	[test file]	[test file]	**Moyenne**
StillOlaf_24	18.9	14.45	16.68
StillOlaf_25	18.76	15.24	17.00
StillOlaf_26	18.84	15.56	17.20
StillOlaf_27	19.29	15.72	17.51
StillOlaf_28	19.35	15.53	17.44
Moyenne	19.028	15.3	17.16

TABLE 3.7 Scores BLEU pour les modèles génériques & modèles spécialisés (discours oral) pour les fichiers de test du discours oral

Modèles Generiques				
Model ID	[test file]	[test file]	[test file]	Moyenne
StillOlaf 01	38.11	32.47	30.51	33.70
StillOlaf 02	19.3	31.67	28.41	26.46
StillOlaf 03	43.91	39.74	36.82	40.16
StillOlaf 04	49.38	41.3	38.86	43.18
StillOlaf 05	50.3	40.64	39.87	43.60
StillOlaf 06	48.3	41.72	39.36	43.13
StillOlaf 07	44.96	41.42	38.96	41.78
StillOlaf 08	51.79	42.86	41.38	45.34
StillOlaf 09	47.26	42.77	38.63	42.89
StillOlaf 10	51.44	42.68	40.5	44.87
StillOlaf 11	52.44	42.79	41	45.41
StillOlaf 12	50.75	43.66	40.48	44.96
StillOlaf 13	45.88	40.96	36.5	41.11
StillOlaf 14	51.2	42.61	38.8	44.20
StillOlaf 15	52.62	43.56	41.04	45.74
StillOlaf 16	50.88	43.12	40.97	44.99
StillOlaf 17	49.95	41.82	38.38	43.38
StillOlaf 18	54.23	43.75	42.01	46.66
StillOlaf 19	52.73	43.86	41.05	45.88
StillOlaf 20	54.39	43.74	41	46.38
StillOlaf 21	53.76	44.01	41.42	46.40
StillOlaf 22	50.39	43.7	38.84	44.31
StillOlaf 23	55.62	44.23	41.89	47.25
StillOlaf 24	54.25	44.2	41.67	46.71
StillOlaf 25	54.05	43.74	41.52	46.44
StillOlaf 26	54.84	44.51	41.48	46.94
StillOlaf 27	53.58	44.43	41.4	46.47
StillOlaf 28	51.76	44.35	40.17	45.43
StillOlaf 29	54.18	44.37	41.16	46.57
StillOlaf 30	55.62	44.52	42.1	47.41
Moyenne	49.93	42.31	39.54	43.93
Modèles Spécialisés				
Model ID	[test file]	[test file]	[test file]	Moyenne
StillOlaf_31	56.23	44.73	42.35	47.77
StillOlaf_32	56.24	44.3	42.97	47.84
StillOlaf_33	54.02	43.61	39.47	45.70
StillOlaf_34	57.17	44.29	42.55	48.00
StillOlaf_35	56.66	44.54	42.11	47.77
Moyenne	56.06	44.29	41.89	47.42

TABLE 3.8 Scores BLEU pour les modèles génériques& modèles spécialisés (juridique) pour les fichiers de test du domaine juridique

Après l'observation des résultats, de toute évidence, les modèles spécialisés sont toujours plus performants que les modèles génériques. Par exemple, la meilleure micro-moyenne concernant les fichiers du domaine d'informatique calculée d'après les modèles spécialisés augmente 8,9 points BLEU par rapport à celle de modèles génériques.

3.3 Partie II : Calcul de la similarité

Dans cette partie, nous détaillerons les expériences de classification automatique des textes. Comme décrit en section 2.4, la classification de textes constitue une étape intermédiaire pour générer une traduction pertinente. Dans notre expérimentation, la

classification repose sur le texte en langue source, l'objectif étant de détecter le domaine du texte d'entrée afin de produire sa traduction en langue cible en utilisant le modèle de traduction du domaine prédit par le classifieur.

3.3.1 Corpus

Les données du corpus d'entraînement pour la tâche de classification proviennent des 32 paires de documents préparés précédemment. Comme la classification s'appuie sur le texte en langue source, nous ne traitons que les documents français.

Pour chaque paire de documents, nous avons nettoyé les lignes vides, éliminé les lignes en double et effectué une permutation aléatoire des lignes afin d'obtenir des données propres. Puis pour chaque domaine, nous avons concaténé les documents du domaine et les réécrivons dans un nouveau fichier.

Concernant l'échantillonnage des données pour le corpus d'entraînement, nous avons d'abord extrait 10 000 lignes par domaine pour constituer 20 nouveaux documents par domaine (informatique, discours oral et juridique), chaque document contenant 500 lignes. Pour le corpus de test, nous employons les fichiers en langue source du corpus de test conçus pour la tâche de traduction qui concernent ces 3 domaines.

Les fichiers de test des autres domaines ne seront pas pris en compte dans cette expérience du fait du manque des données pour entraîner des modèles de traduction spécialisés.

Sachant qu'une grande partie de phrases parallèles sont obtenues via un crawl, elles sont restructurées dans les documents de manière artificielle; en outre, les documents sont normalement très volumineux, nous avons choisi de restructurer les lignes dans les documents, afin de rendre les valeurs IDF significatives (l'entraînement du classifieur Naïve Bayes suppose une représentation de valeurs TDF-IDF comme features).

3.3.2 Classification

3.2.3.1 Expérience avec KenLM

Nous entraînons d'abord un modèle de langue avec les 10 000 lignes extraites grâce à l'outil KenLM. Pour chaque domaine, nous concaténons les documents du domaine dans un nouveau fichier. Vu qu'il demande un texte tokenisé en entrée, nous utilisons une bibliothèque python NLTK qui applique une tokenisation au niveau des mots. Ensuite, une chaîne de prétraitements est appliquée pour normaliser la casse et les caractères accentués. Enfin, nous entraînons un modèle de langue de 3-grammes par domaine.

3.2.3.2 Expérience avec Naïve Bayes

Le deuxième classifieur entraîné repose sur l'algorithme de la loi multinomiale de Naive Bayes qui nécessite une représentation vectorielle de valeurs TF-IDF comme features du texte. Plusieurs prétraitements ont été appliqués sur le texte d'entraînement. Pour le corpus d'entraînement et de test, il s'agit de supprimer les mots grammaticaux, de normaliser la casse et les lettres accentuées et d'effectuer une tokenisation en mots. Nous les transformons ensuite en deux représentations vectorielles de valeurs de TF-IDF (l'une pour le corpus d'entraînement, l'autre pour le corpus du test). Enfin nous entraînons un classifieur Naïve Bayes fondé sur la représentation vectorielle du corpus d'entraînement.

3.3.3 Évaluation

Nous avons entraîné deux classifieurs avec un corpus d'entraînement constitué de 10 000 lignes. Les méthodes d'évaluation sélectionnées pour les deux classifieurs sont précision, rappel et F-mesure. Nous calculons la moyenne pondérée pour chaque composant de ces derniers puisque la distribution de domaines est déséquilibrée dans le corpus de test.

Domaine	Précision	Rappel	F-mesure	Nombre d'instances du domaine
Informatique	1,00	0,80	0,89	5
Discours oral	0,67	1,00	0,80	2
Juridique	1,00	1,00	1,00	3
Moyenne pondérée	0,93	0,90	0,90	10

TABLE 3.9 Évaluation du modèle de langue entraîné avec un échantillon de 10 mille lignes

Domaine	Précision	Rappel	F-mesure	Nombre d'instances du domaine
Informatique	1,00	1,00	1,00	5
Discours oral	1,00	1,00	1,00	2
Juridique	1,00	1,00	1,00	3
Moyenne pondérée	1,00	1,00	1,00	10

TABLE 3.10 Évaluation du classifier Naïve Bayes entraîné avec un échantillon de 10 mille lignes

Une évaluation supplémentaire reposant sur la K-fold validation croisée (K-fold cross-validation) a été réalisée pour le classifieur Naïve Bayes. Le principe de cette méthode est de diviser l'échantillon d'origine en k échantillons, puis de sélectionner l'un de ces k échantillons comme ensemble de validation et les k-1 autres échantillons constituant l'ensemble d'apprentissage. On calcule la précision, le rappel et la F-mesure, puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les k-1 échantillons qui n'ont pas encore été utilisés pour la validation du modèle [***ref. Wikipédia***[24]].

Nous divisons le corpus d'entraînement en 6 échantillons étant donné qu'il y a 60 documents dans le corpus d'entraînement à raison de 20 documents par classe. Le corpus de validation contient 10 documents par expérience, nous calculons sa précision, rappel et F-mesure.

Pour toutes les expériences de validation croisée, le résultat sur précision, rappel et F-mesure est toujours de 100%. Il est possible que les données d'entraînement dans cette tâche de classification sont proches de celles du test, vu qu'il vient du même corpus. Une autre possibilité est que la tâche de classification est plutôt simple du fait que le nombre de documents dans le corpus de test et de validation est petit.

Conformément aux résultats exposés précédemment, d'un côté le classifieur Naïve Bayes produit une meilleure performance pour la classification des documents dans le corpus de test. De l'autre côté, le modèle de langue récolte une F-mesure de 90%, il montre son insuffisance pour classification des documents de domaine discours oral.

3.4 Partie III : De la classification à la traduction

Nous avons entraîné deux classifieurs, ils ont pour but de prédire le domaine d'un texte donné de sorte que nous pouvons lui attribuer une traduction générée par nos modèles de traduction.

Dans cette partie, nous adaptons les sorties de classifiers à la tâche de traduction. Les expériences reposent sur deux hypothèses. La première est basée sur l'ensemble de document à traduction, tandis que la deuxième est que toutes les phrases n'appartiennent pas forcément au même domaine que celui de son document superordonné.

3.4.1 Traduction basée sur la classification

3.4.1.1 Au niveau du document

Le classifier Naïve Bayes montre sa robustesse (100% de F-mesure) lors de la prédiction des classes pour les documents en source dans le corpus de test. D'un point de

24https://fr.wikipedia.org/wiki/Validation_crois%C3%A9e

vue macro, nous voulons d'abord prédire le domaine du document, puis utiliser le modèle de traduction spécialisé du même domaine afin d'obtenir la traduction entière.

3.4.1.2 Au niveau de la phrase

Les textes de ces 3 domaines sont très représentatifs, par exemple les documents du domaine informatique comportent souvent des équations mathématiques ou des caractères spéciaux. Cependant, certaines phrases peuvent porter des caractéristiques des autres domaines.

Lors de l'entraînement des modèles de traduction, les traductions sont générées automatiquement par le système : chaque modèle de traduction produit des traductions pour tous les documents du corpus de test.

Pour nos modèles spécialisés, chaque ligne du fichier de traduction en sortie se compose d'un score de confiance et la traduction de cette phrase, séparés par « ||| », comme présenté dans le tableau 3.11.

Source	Traduction
53. Rappelle le paragraphe 47 de sa résolution 56/64 B dans lequel elle a décidé, compte tenu de la réussite du projet pilote sur l'élaboration d'une capacité de radiodiffusion internationale pour l'Organisation des Nations Unies et de l'importance de la distribution de ses programmes et des partenariats établis, d'augmenter la capacité de radiodiffusion internationale de l'Organisation dans les six langues officielles ;	-0.088587 ||| 53. 回顾其第 56/64 B 号决议第 47 段,其中大会决定,鉴于关于发展联合国国际无线电广播能力的试点项目的成功以及其节目的分发和伙伴关系的重要性,加强联合国六种正式语文的国际无线电广播能力;
8. Prie instamment les États Membres de prendre aux niveaux international et national des mesures efficaces pour prévenir et combattre le trafic de biens culturels, notamment par une formation spéciale des services frontaliers, des douanes et de police ;	-0.147386 ||| 8. 敦促会员国采取有效国际和国家措施,防止和打击贩运文化财产,包括对边境服务、海关和警察进行特别培训;
9. Demande aux États de mettre en place, compte tenu de la Convention sur la diversité biologique, des programmes nationaux, régionaux et internationaux pour contrecarrer l'appauvrissement de la biodiversité marine dans la mer des Caraïbes, en particulier d'écosystèmes fragiles comme les récifs coralliens ;	-0.088275 ||| 9. 吁请各国考虑到《生物多样性公约》,制订国家、区域和国际方案,制止加勒比海海洋生物多样性的丧失,特别是珊瑚礁等脆弱生态系统的丧失;
Titre III	-0.168275 ||| 第三编
Déplorant les difficultés grandissantes que rencontre la presse pour s'exprimer librement depuis les graves incidents d'avril 2000,	-0.189016 ||| 痛惜自 2000 年 4 月发生严重事件以来,媒体在自由表达意见方面日益困难,

TABLE 3.11 Extrait de phrases source

avec leurs traductions en cible traduites par le modèle spécialisé juridique

D'un point de vue microscopique, l'expérience nécessite un classifieur multi-classe qui calcule la probabilité de chaque phrase pour chaque classe du modèle. La probabilité portée sur chaque classe sera alors utilisée comme le poids de cette classe. Pour chaque modèle spécialisé, il s'agit d'extraire la meilleure traduction (score BLEU le plus élevé) pour tous les documents en source.

La pondération s'applique uniquement au niveau de phrase. Pour chaque phrase du document source, nous calculons d'abord la probabilité qu'il appartienne à chaque domaine, puis multiplions cette probabilité par le score de confiance de la traduction. Ensuite nous écrivons la traduction portant le meilleure score pondéré dans le fichier de sortie. De cette manière, nous obtenons finalement une traduction synthétique pour le document entier. Le processus de cette étape est schématisé dans la Figure 3.3.

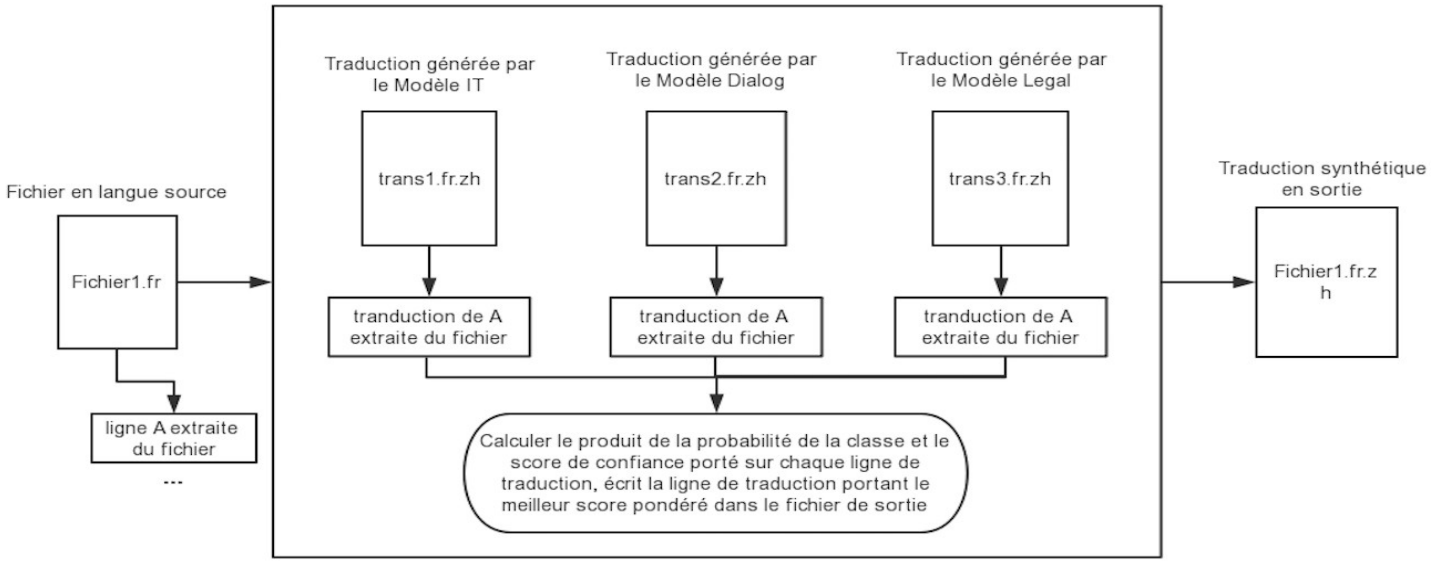

FIGURE 3.3 Schéma d'adaptation de classification à traduction

Dans un premier temps nous utilisons le modèle de langue et le classifier Naïve Bayes entraîné avec 10 000 lignes. La difficulté avec le classifieur Naïve Bayes est que, généralement, la probabilité de chaque classe n'est pas très loin de l'une à l'autre. Nous ne pouvons pas assigner un domaine à une phrase du à une faible différence dans leurs probabilités. En second lieu nous employons le modèle de langue pour calculer la perplexité pour chaque domaine, ensuite nous procédons le processus de pondération.

Étant donné que le modèle de langue entraîné avec 10 000 échantillons n'a pas obtenu une F-mesure élevée lors de l'évaluation sur le corpus du test. Nous essayons

d’enrichir le corpus d’entraînement afin d’obtenir un modèle de classification robuste. Lorsque la quantité d'échantillon de données atteint 20 mille ou plus, la F-mesure du modèle de langue passe de 90% à 100%.

Inévitablement, une phrase peut ne pas appartenir à aucune des 3 classes prédéfinies étant donné que les classes sont indépendantes et assez distinctes. Nous envisageons de définir un seuil qui permettra de distinguer des données de classes prédéfinies et des données hors domaine, pour la traduction de ces dernières nous adoptons le modèle de traduction générique.

Nous posons notre première hypothèse : pour une phrase donnée, si sa perplexité pour chaque classe dépasse la perplexité moyenne de la classe, nous considérons qu’il s’agit d’une phrase hors-domaine. Par conséquent, le seuil sera défini comme la perplexité moyenne de chaque classe.

Pour la deuxième hypothèse, nous supposons que toutes les données de la classe constituent une distribution normale. Nous définissons un intervalle pour chaque classe égale à la perplexité moyenne de la classe plus moins (±) l’écart-type. Les données en dehors de cet intervalle seront considérées comme hors domaine.

3.4.2 Évaluation

Les précédentes expériences ont pour but de générer de traductions pertinentes pour les documents du corpus de test (dans nos expériences). Nous avons utilisé une méthode hybride en utilisant un classifieur et la spécialisation des modèles de traduction en fine-tuning. L'évaluation de nos méthodes est faite via la métrique BLEU.

Le meilleure score BLEU pour les documents du domaine correspondant au modèle spécialisé dépasse celle correspondant au modèle générique (baseline). Nous avons gagné 8,95 points pour les documents du domaine d’informatique, 1,96 pour le domaine de discours oral, et 0,59 pour le domaine juridique.

Malgré que les domaines des documents soient connus du système pendant la traduction, quand il s’agit de nouvelles données entrées dans le corpus de test, nous pouvons prédire la classe des documents et ainsi sélectionner les modèles correspondants.

Dans cette section, nous évaluons d’abord la méthode de pondération, puis comparons cette dernière avec la méthode de pondération des scores.

	Modèles Spécialisés												
	IT					Dialog		Legal			Moyenne		
Model ID	[test file]	[test file]	[test file]	[test file]	[test file]	[test file]	[test file]	[test file]	[test file]	[test file]	Micro_IN_domain	Micro_OUT_domain	Macro
IT_StillOlaf_31	43.60	58.22	45.39	41.77	38.60	14.74	8.12	40.37	37.76	36.28	45.52	27.45	36.49
IT_StillOlaf_32	44.76	59.29	47.59	42.62	39.32	14.07	7.66	32.46	33.19	30.92	46.72	23.66	35.19
IT_StillOlaf_33	45.70	60.44	48.40	43.27	39.93	12.38	6.69	26.81	30.08	28.19	47.55	20.83	34.19
IT_StillOlaf_34	45.90	60.49	49.00	43.57	40.19	12.78	6.36	30.75	28.63	27.38	47.83	21.18	34.51
IT_StillOlaf_35	47.01	61.42	49.08	44.58	41.7	12.45	6.74	24.33	26.72	25.29	48.76	19.11	33.93
Dialog_StillOlaf_24	34.66	42.89	32.77	30.36	28.83	18.9	14.45	46.17	39.84	37.52	16.68	36.63	32.64
Dialog_StillOlaf_25	33.61	41.79	31.43	29.56	28.55	18.76	15.24	47.8	37.68	36.23	17.00	35.83	32.07
Dialog_StillOlaf_26	32.66	41.68	30.9	29.32	28.01	18.84	15.56	48.53	37.76	35.35	17.20	35.53	31.86
Dialog_StillOlaf_27	31.44	38.31	29.08	27.62	26.6	19.29	15.72	43.45	36.24	33.52	17.51	33.28	30.13
Dialog_StillOlaf_28	30.11	36.79	27.41	26.8	25.67	19.35	15.53	46.03	35.29	32.9	17.44	32.63	29.59
Legal_StillOlaf_31	34.35	42.39	32.25	31.85	17.14	17.14	10.2	56.23	44.73	42.35	47.77	26.47	32.86
Legal_StillOlaf_32	33.1	41.49	32.29	30.17	16.85	16.85	9.85	56.24	44.3	42.97	47.84	25.80	32.41
Legal_StillOlaf_33	30.58	34.51	27.77	27.73	16.29	16.29	9.27	54.02	43.61	39.47	45.70	23.21	29.95
Legal_StillOlaf_34	32.13	38.41	29.12	29.17	16.44	16.44	9.5	57.17	44.29	42.55	48.00	24.46	31.52
Legal_StillOlaf_35	30.53	36.21	27.8	28.41	16.32	16.32	9.24	56.66	44.54	42.11	47.77	23.55	30.81
											Synthetic_moyenne	Best_synt_moyenne	Best_spec_moyenne
10k_synthetic_trans_baseline	46.85	61.74	48.88	44.1	40.86	18.56	13.21	57.21	44.24	42.46	41.81	42.13	42.331
10k_synthetic_trans_threshold_avr	43.28	57.92	44.11	40	38.55	18.38	13.23	55.81	44.43	41.96	39.77		
10k_synthetic_trans_distribN	44.81	56.7	45.74	41.18	38.76	18.75	11.96	55.99	43.95	42.05	39.99		
20k_synthetic_trans_baseline	46.88	61.77	49.03	44.26	41.19	18.45	13.94	57.16	44.5	42.38	41.96		
20k_synthetic_trans_threshold_avr	43.03	57.91	44.5	40.54	38.76	18.34	13.81	55.83	44.49	41.98	39.92		
20k_synthetic_trans_distribN	44.86	56.78	45.85	41.73	39.14	18.72	12.46	55.97	44.25	42.13	40.19		
100k_synthetic_trans_baseline	46.73	61.77	49.12	44.48	41.48	18.5	14.24	57.16	44.79	42.55	42.08		
100k_synthetic_trans_threshold_avr	43.18	57.91	44.36	42.07	39.36	18.6	14.14	55.96	44.46	42.09	40.21		
100k_synthetic_trans_distribN	45.03	57.24	46.17	43.31	40.15	18.8	12.48	56.21	44.38	42.09	40.59		

TABLE 3.12 Scores BLEU pour les modèles spécialisés et la méthode de pondération des scores

Le tableau 3.12 présente les scores BLEU de traduction pour tous les documents du corpus de test. Les traductions sont fournies par les modèles spécialisés du domaine d'informatique, juridique et de discours oral, et par nos expériences de pondération.

Dans un premier temps nous évaluons nos méthodes de pondération. Les scores BLEU pour la méthode de pondération sont présentés dans la partie inférieure du tableau. Il s'agit de 3 séries d'expériences basées sur un échantillon de 10 mille, 20 mille et 100 mille données respectivement, Chaque série d'expériences consiste en une baseline (sans introduction d'un seuil), une expérience avec la moyenne de perplexité du domaine comme seuil, ainsi qu'une expérience qui prend la moyenne plus ou moins l'écart type comme l'intervalle du seuil.

D'après les scores BLEU calculés, l'expérience de pondération sans l'introduction du seuil acquit un meilleur résultat que les expériences avec un seuil prédéfini. Nous constatons que la perplexité calculée pour chaque phrase varie selon la similarité entre des features de la phrase et celles du corpus d'entraînement, mais aussi en fonction de la longueur de phrase. Pour les futurs travaux, nous devons explorer une méthode permettant de normaliser la perplexité de la phrase et celle du seuil, afin de diminuer le taux de faux positifs et de faux négatifs.

En second lieu, nous comparons les méthodes de pondération avec la méthode de spécialisation en fine-tuning. Globalement, les scores BLEU obtenus par la spécialisation sont plus élevés que ceux acquis par la méthode de pondération.

D'un point de vue macro, nous calculons la moyenne de scores BLEU pour chaque expérience parmi les 3 séries d'expériences de pondération, puis nous comparons la moyenne la plus élevée avec celle pour la traduction de chaque document du domaine généré par les modèles spécialisés. La moyenne pour ces derniers correspond aux scores BLEU des traductions des documents du domaine (présenté avec un arrière-plan vert foncé

dans le tableau 3.12). Nous avons finalement une meilleure moyenne de 42,08 BLEU pour les méthodes de pondération, et 42,33 pour la spécialisation.

D'un point de vue microscopique, nous calculons la moyenne des scores BLEU les plus élevés issus de la méthode pondération pour chaque document (présenté avec un fond rose dans le tableau). Nous avons une micro-moyenne de 42,13 en score BLEU par rapport à 42,33 pour la spécialisation.

D'après les résultats présentés, la méthode de spécialisation présente un gain subtil (moins d'un point en BLEU) par rapport à la méthode de pondération. La méthode de pondération prouve sa disponibilité et son efficacité dans la tâche de traduction : elle a dépassé la baseline (La moyenne la plus élevée est 34,26) de 7,87 en score BLEU. Cependant, nous ne pouvons pas établir qu'elle est moins performante que la méthode de spécialisation, étant donné la différence minuscule entre les scores. En outre, les classifieurs sont entraînés par les données de classes relativement distinctes, la relation entre classes n'est pas nécessairement évidente.

Il s'agit des limitations pour nos expériences. Premièrement, le nombre des catégories de données dans le corpus de test est assez petit. Ce problème vient du manque de corpus parallèle pour français-chinois. Dans notre corpus d'entraînement, seulement les données de ces 3 classes sont suffisantes pour procéder à une spécialisation des modèles, et il est difficile de trouver de nouvelles données ayant une référence pour le corpus de test.

Deuxièmement, les scores BLEU pour la spécialisation et la méthode de pondération sont assez proches, il est difficile de choisir un meilleur modèle en se basant uniquement sur une seule métrique. Une évaluation humaine peut être une possibilité pour résoudre ce problème, toutefois cela demande énormément de temps pour l'évaluation et il dépend du jugement subjectif de l'évaluateur. Au milieu professionnel, nous avons environ 140 paires de langues mais beaucoup de langues pour lesquelles nous n'avons pas de locuteur dans l'équipe (eg. en géorgien).

4. Conclusion

Tout au long de nos expériences, nous avons d'abord entraîné un modèle de traduction générique avec les données issues de tous les domaines du corpus d'entraînement. Dans un deuxième temps, nous avons sélectionné les données des 3 domaines prédéfinis consistant le domaine d'informatique, juridique et de discours oral, et continuer d'entraîner 3 modèles de traduction spécialisés à partir de la modèle générique (spécialisation en fine-tuning). L'entraînement des modèles est via le système de traduction PN9 implémenté par Systran. À la fin de chaque itération d'entraînement, la traduction pour tous les documents du corpus de test est générée automatique par le système.

Ensuite il s'agit d'implémenter des classifieurs (un modèle de langue et un classifieur Naïve Bayes) ayant deux fonctionnalités : nous pouvons s'en servir pour prédire le domaine du document à traduire, afin de pouvoir sélectionner un modèle adéquat et obtenir une traduction pertinente pour ce document. Pour chaque phrase du document, le classifieur calcule la probabilité qu'elle appartienne à chaque classe prédéfinie. La probabilité est utilisée comme un poids sur le score de confiance associé à la traduction pour chaque phrase, puis nous reconstituons une traduction phrase par phrase en choisissant la traduction portant le meilleur score pondéré.

L'évaluation des classifieurs a été réalisée par la classification des documents en langue source issus du corpus de test de traduction. Avec 10 mille de données d'entraînement, nous obtenons 90% en F-mesure pour le modèle de langue, et 100% pour le classifieur Naïve Bayes. Nous avons obtenu des bonnes F-mesure pour la classification des documents, néanmoins, il s'avère difficile quand il s'agit uniquement d'une phrase. D'un côté, la classification des documents est plutôt facile. À la base, ce sont des fichiers qui ont été splittés et mélangés dans le corpus d'entraînement et le corpus du test. En conséquence, les données se ressemblent beaucoup. Il se reflète probablement sur leur vocabulaire en commun. En outre, en fonction de volume de données disponibles, nous avons choisi 3 catégories qui sont assez différentes les unes des autres. Par ailleurs, il est plus difficile de classifier une phrase. La phrase porte beaucoup moins d'informations, surtout quand nous éliminions les mots grammaticaux, et ainsi les features Tfidf sont moins fiables pour une grande partie du corpus.

L'évaluation de méthodes de traduction a été effectuée en calculant la moyenne de scores BLEU pour tous les documents du corpus de test. Selon les résultats obtenus, la spécialisation et la méthode de pondération recueillent des bons résultats, la meilleure moyenne de scores BLEU calculée pour la méthode de pondération est 42,13 tandis qu'il s'agit de 42,33 BLEU pour la spécialisation. Ces deux méthodes ont été prouvées ses efficacités, elles sont plus performantes que la baseline (traduction avec un modèle générique). Dans les futurs travaux, l'enrichissement des catégories et la quantité de corpus parallèle seront mise en pratique. Nous appliquerons aussi une intervention humaine pour comparer avec les résultats obtenus via scores BLEU.

Bibliographie

Wu et al.,2016 Google's Neural Machine Translation System: Bridging the Gap between Human and Machine translation https://arxiv.org/pdf/1609.08144.pdf
Georgetown University and IBM, 1954 The first public demonstration of machine translation:the Georgetown-IBM system, 7th January 1954
P. Brown et al., 1993 The mathematics of statistical machine translation: parameter estimation. Computational Linguistics, 19(2), 263-311.
Cho, Bengio et al, 2014 Neural Machine Translation by Jointly Learning to Align and Translate
Wu et al.,2016 Google's Neural Machine Translation System: Bridging the Gap between Human and Machine translation Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
Bahdanau et al., 2015 Neural Machine Translation By Jointly Learning To Align And Translate
Vaswani et al., 2017 Attention Is All You Need
Boxing Chen et al., 2017 Cost Weighting for Neural Machine Translation Domain Adaptation
Klein et al., 2018 Open-Source Toolkit for Neural Machine Translation
Servan et al., 2016 Domain specialization: a post-training domain adaptation for Neural Machine Translation
Cho et al., 2014 On the Properties of Neural Machine Translation: Encoder-Decoder Approaches
Zalmout et al,. 2017 Optimizing Tokenization Choice for Machine Translation across Multiple Target Languages
Sennrich et al,. 2016 Neural Machine Translation of Rare Words with Subword Units Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1715–1725, Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics
Kudo et al,. 2018 SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing
Sen et al,. 2016 Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.
Simard, 1998 Bilingual Sentence Alignment: Balancing Robustness and Accuracy
Kenneth Heafield, 2011 KenLM: Faster and Smaller Language Model Queries

Annexe

A. Script pour le prétraitement des données pour la tâche de classification

```
class Preprocess(object):
    # dir = train_dir or tst_dir
    def __init__(self, dir):
        self.dir=dir

    def docref(self):
        # return a tuple which contains filename and reference(domaine of file)
        pat_it=re.compile(r'(.*)IT(.*)\.fr')
        pat_dialog=re.compile(r'(.*)Dialog(.*)\.fr')
        pat_legal=re.compile(r'(.*)Legal(.*)\.fr')
        file_ref=[]
        path, dirs, files=next(os.walk(self.dir))
        #it=0, dialog=1, legal=2
        for file in os.listdir(self.dir):
            if pat_it.match(file):
                file_ref.append((file,0))
            elif pat_dialog.match(file):
                file_ref.append((file,1))
            elif pat_legal.match(file):
                file_ref.append((file,2))
            else:continue
        return file_ref

    def corpusref(self):
        # write the tokenized and normalized sentences of each doc in a list, and the doc ref in an array
        file_ref_list=self.docref()
        Y = np.zeros((len(file_ref_list),1))
        lines=''
        corp=[]
        i=0
        for file,ref in file_ref_list:
            #it,dialog,legal=0,1,2
            Y[i,0]=ref
            sentences=self.normalize_accent(file)
            corp.append(sentences)
            i+=1
        return corp,Y

    def normalize_accent(self,file):
        # load tokenizer, return the tokenized and normalized sentences in current file
        tok=Tokenizer(self.dir+'/'+file)
        tokenized_file=tok.tokenize()
        sentences=""
        for tokenized_line in tokenized_file:
            for word in nltk.wordpunct_tokenize(tokenized_line):
                word=unidecode.unidecode(word)
                sentences+=word.lower()+' '
        return sentences

```

B.1.1 Script pour l'entraînement du modèle de langue

```
#!/bin/bash
# kenLM model training

DIR=$1 #raw data, eg. /home/yunbei/Documents/workdata/memoire/data/training/clf_dist_10k_average
concat_txt_DIR=$2 #concatenate data to path, eg. /home/yunbei/Documents/workdata/memoire/data/training/clf_concat_10k
MODELS=$3 #save model to path, eg. /home/yunbei/Documents/workdata/memoire/clf/ngram_model_10k

lmpath=$4 #clf/ngrams/tst_kenLM/kenlm/bin
mkdir $concat_txt_DIR
mkdir $MODELS
start=$SECONDS

function training(){
    for file in `ls $concat_txt_DIR`
    do
        # if [[ $file == *"IT"*.{fr,zh} ]];then
        if [[ $file == *"IT"*.fr ]];then
            `python3 scripts/tokenizer.py $concat_txt_DIR/$file | $lmpath/lmplz -o 3 > $MODELS/IT.arpa`
        fi
        if [[ $file == *"Dialog"*.fr ]];then
            `python3 scripts/tokenizer.py $concat_txt_DIR/$file| $lmpath/lmplz -o 3 > $MODELS/Dialog.arpa`
        fi
        if [[ $file == *"Legal"*.fr ]]; then
            `python3 scripts/tokenizer.py $concat_txt_DIR/$file| $lmpath/lmplz -o 3 > $MODELS/Legal.arpa`
        fi
    done
    list="IT Dialog Legal"
    for item in $list
    do
        $lmpath/build_binary $MODELS/$item".arpa" $MODELS/$item.klm
    done
}

cat $DIR/*"IT"*.fr > $concat_txt_DIR/IT.fr
cat $DIR/*"Legal"*.fr > $concat_txt_DIR/Legal.fr
cat $DIR/*"Dialog"*.fr > $concat_txt_DIR/Dialog.fr
training $1 $2 $3 $4

time=$(( SECONDS - start ))
echo $time
```

```
class KenLM(object):
    def __init__(self, model_dir, tst_dir):
        self.model_dir=model_dir #path of directory containing the kenLM model
        self.tst_dir=tst_dir #path of test set directory

    def ref(self,list_files):
        #get un array of refs for the list of file names given
        s=0
        y_true=np.zeros((len(list_files),1))
        #cl_IT=0, cl_Dialog=1, cl_Legal=2
        for i in range(len(list_files)):
            if re.match(r'(.*)IT(.*)',list_files[i]):s=0
            if re.match(r'(.*)Dialog(.*)',list_files[i]):s=1
            if re.match(r'(.*)Legal(.*)',list_files[i]):s=2
            y_true[i,0]=s
        return y_true

    def best_row_score(self,list_files,array):
        #return the best one of three perplexity scores
        y_pred=np.zeros((len(list_files),1))
        list_pred=np.argmax(array,axis=1)
        #print(array)
        for j in range(len(list_pred)):
            y_pred[j,0]=list_pred[j]
        return y_pred

    def load_kenLM_model(self):
        #load the trained kenLM models with the python API of kenlm
        IT_m=kenlm.LanguageModel(self.model_dir+'/IT.klm')
        Legal_m=kenlm.LanguageModel(self.model_dir+'/Legal.klm')
        Dialog_m=kenlm.LanguageModel(self.model_dir+'/Dialog.klm')
        return IT_m,Dialog_m,Legal_m

    def calcul_perplexity(self):
        pat_it=re.compile(r'(.*)IT(.*)\.fr')
        pat_dialog=re.compile(r'(.*)Dialog(.*)\.fr')
        pat_legal=re.compile(r'(.*)Legal(.*)\.fr')

        #initialize an array to store the predicted perplex value
        init=np.array([[0,0,0]])
        np.set_printoptions(threshold=np.nan) # show table details
        docs_perlexity={}
        list_files=[]

        IT_m,Dialog_m,Legal_m=self.load_kenLM_model() #get trained kenLM models

        for file in os.listdir(self.tst_dir):
            sentences=[]
```

```
            tok = Tokenizer(self.tst_dir+'/'+file) # tokenize a text file in test set
            sentences=tok.tokenize() #list of all tokenized sentences of file)

            # initialize perplexity value for each class
            score_IT, score_Dialog, score_Legal=0.0, 0.0, 0.0
            doc_perlexity=[]

            if pat_it.match(file) or pat_legal.match(file) or pat_dialog.match(file):
                count=0 # count lines
                for sentence in sentences:
                    #add a sentence perplexity of each class to a tuple
                    sent_perlexity=(IT_m.score(sentence),Dialog_m.score(sentence),Legal_m.score(sentence))

                    #accumulate the perplexity score of each sentence of each class, then compare their average perplexity score
                    score_IT+=IT_m.score(sentence)
                    score_Dialog+=Dialog_m.score(sentence)
                    score_Legal+=Legal_m.score(sentence)
                    count+=1
                    #doc_perlexity.append((sent_perlexity,sentence)) #add perlexities of each sentence to a list
                    doc_perlexity.append(sent_perlexity) #add a 3-tuple of perlexities for each sentence to a list
                #docs_perlexity.append((file,doc_perlexity)) #dict type, key=filename and value=list of perplexities contains sentence and sent

                # a dict takes filename as key and a list of tuple as value
                docs_perlexity[file]=docs_perlexity.setdefault(file, [])+doc_perlexity
                #list of filenames
                list_files.append(os.path.basename(file)) #filenames

                # for the current file, calculate the average of perplexity score of each class (SUM of perplex / nb lines)
                moy_IT, moy_Dialog, moy_Legal=score_IT/count, score_Dialog/count, score_Legal/count #count=nb of lines in file

                #In each column of the matrix, we have it dialog legal in turn
                tmp=np.array([[moy_IT, moy_Dialog, moy_Legal]])
                init=np.r_[init, tmp] #concatenate 2 arrays

        init=np.delete(init,0,0) #delete the 1st row [0,0,0]
        y_true=self.ref(list_files)
        y_pred=self.best_row_score(list_files, init)

        return y_true,y_pred, init, docs_perlexity


```

B.1.2 Script pour l'évaluation du modèle de langue

```
    def evaluation(self):
        y_true,y_pred,avr,docs_perlexity=self.calcul_perplexity()
        error=0
        for m in range(y_true.shape[0]):
            if not y_true[m] == y_pred[m]:
                error+=1
        error_rate=str(error/int(y_true.shape[0]))
        fscore=sklearn.metrics.classification_report(y_true, y_pred, labels=None, target_names=None,sample_weight=None, digits=2)#support=T
        print("Gold stantard : "+str(y_true),'\n')
        print("Prediction :"+str(y_pred),'\n')
        return error_rate,fscore

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='calculate similarity by Language model kenLM')
    parser.add_argument("model_DIR", help="trained model directory") #/home/yunbei/Documents/workdata/memoire/clf/ngram_model_10k
    parser.add_argument("testdata_DIR", help="test data directory") #/home/yunbei/Documents/workdata/memoire/data/tst
    args = parser.parse_args()
    model_DIR=args.model_DIR
    testdata_DIR=args.testdata_DIR

    lm=KenLM(model_DIR,testdata_DIR)
    error_rate,fscore=lm.evaluation()

    print("evaluating...")
    print("error rate : "+error_rate)
    print("precision, recall and f1-score table :")
    print(fscore)
```

B.2.1 Script pour l'entraînement du classifieur Naïve Bayes

```
class Tfidf_object(object):
    """return the vectors representation of corpus"""
    def tfidf_vectorization(self):
        # return a TfidfVectorizer object
        stopwords=STOPWORDS.words('french')
        vectorizer=TfidfVectorizer(encoding="utf-8",decode_error="strict",stop_words=stopwords) #analyzer=stemmed_words, ngram_range=(2, 3)
        return vectorizer


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='preprocess the traing and tst data')
    parser.add_argument("train", help="training data directory") #data/clf_dist_10k_average/
    parser.add_argument("test", help="test data directory") #data/tst/
    parser.add_argument("model_path", help="to save the multiclass classifier to the give path") #clf/$FILENAME
    args = parser.parse_args()
    train_dir=args.train
    tst_dir=args.test
    model_path=args.model_path

    start_time = time.time()
    print("processing the traininig corpus")
    processed_train=Preprocess(train_dir)
    train_x,train_y=processed_train.corpusref() #train_x is a list in which every component present the content of a doc in training set, t

    print("processing the test corpus")
    processed_tst = Preprocess(tst_dir)
    tst_x, tst_y=processed_tst.corpusref()

    print('\n','------------------------------ classification ------------------------------------','\n')
    #clf=SVC(gamma='scale',C=1) #valid=1, tst=0.90
    #clf=LogisticRegression(solver='lbfgs',multi_class='multinomial') #valid=1, tst=0.90
    clf=MultinomialNB() #valid=1, tst=1

    tfidf = Tfidf_object() # an instance of class Tfidf_object
    vectorizer=tfidf.tfidf_vectorization()

    #cross validation
    print("Cross validation evaluating...",'\n')
```

B.2.2 Script pour l'évaluation du classifieur Naïve Baye

```
    #cross validation
    print("Cross validation evaluating...",'\n')

    """
    input : this method takes a list in which every component present the content of a doc in training set, doc ref of training set, classfier désired
    output : precision, rappel, F-score table
    """
    def cross_validation(train_x,train_y,classifier):
        skf = StratifiedKFold(n_splits=6,shuffle=True,random_state=0) #stratified K-Folds cross-validator
        for train_index, valid_index in skf.split(train_x, train_y): #split the training set to sub training set and validation set
            # train_index, valid_index are the index of each element (doc content) of list
            X_train=[]
            X_valid=[]
            y_train=np.zeros((len(train_index),1))#initialise array to store the refs
            y_valid=np.zeros((len(valid_index),1))
            i=0
            for indice in list(train_index):
                X_train.append(train_x[indice])#append corresponding doc content in list
                y_train[i]=train_y[indice]#append corresponding doc ref in array
                i+=1
            j=0
            for index in list(valid_index):
                X_valid.append(train_x[index])
                y_valid[j]=train_y[index]
                j+=1

            # matrix of tfidf vectors representation
            X_traintfidf = vectorizer.fit_transform(X_train).toarray()
            X_validtfidf = vectorizer.transform(X_valid).toarray()

            # normalize y_train, y_valid to a regular format array in sklearn
            y_train=np.array(y_train.ravel()).astype(int)
            y_valid=np.array(y_valid.ravel()).astype(int)

            # round 1-10
            classifier.fit(X_traintfidf, y_train) #fit the classifier according to the given sub training data
            y_pred = classifier.predict(X_validtfidf) #perform classification on samples in X_validtfidf
            print("gold stantard : ",y_valid)
            print("prediction : ",y_pred)
            print(classification_report(y_valid, y_pred, labels=None, target_names=None,sample_weight=None, digits=2))
            print("**************************************************",'\n')

    cross_validation(train_x,train_y,clf)
```

C. Script pour l'extraction de la traduction et score de confiance

```
#!/bin/bash
# kenLM model training

spec_infer_DIR=$1 #/home/yunbei/Documents/workdata/memoire/mt/trans_nbestscores/
best_spec_infer_DIR=$2 #/home/yunbei/Documents/workdata/memoire/mt/best_spectrans_inferences/

# path of indomain documents with best translation
Pat_it_ind_bests="(^(35).*IT.*fr\.zh)"
Pat_it_ood_bests="(^(31).*((Dialog)|(Legal)).*fr\.zh)"
Pat_dialog_ood_bests="(^(24).*IT.*fr\.zh)"
it_bests="($Pat_it_ind_bests|$Pat_it_ood_bests)"
dialog_bests="($Pat_dialog_ood_bests|((28-btxt_2dir_fr-XX-zh-YY_Dialog__27843-ted-talks-fbk-r3
legal_bests="(((31-btxt_2dir_fr-XX-zh-YY_IT__27824-TAUS-HW-StrDoc-Download-7742-r2)|(31-btxt_2

for dir in `ls $spec_infer_DIR`
do
    if [[ $dir == "IT" ]];then
        for file in `ls $spec_infer_DIR/$dir`
        do
            if [[ $file =~ $it_bests ]];then
                `cp $spec_infer_DIR/$dir/$file $best_spec_infer_DIR"/IT"`
            fi
        done
    fi
    if [[ $dir == "Dialog" ]];then
        for file in `ls $spec_infer_DIR/$dir`
        do
            if [[ $file =~ $dialog_bests ]];then
                `cp $spec_infer_DIR/$dir/$file $best_spec_infer_DIR"/Dialog/"`
            fi
        done
    fi
    if [[ $dir == "Legal" ]];then
        for file in `ls $spec_infer_DIR/$dir`
        do
            if [[ $file =~ $legal_bests ]];then
                `cp $spec_infer_DIR/$dir/$file $best_spec_infer_DIR"/Legal/"`
            fi
        done
    fi
done
```

D. Script pour calculer le score pondéré

```
    def get_cscore_trans(self):
        ''' return a dictionary with filename as key and a tuple type as value.
        Each element in tuple is a list of confidence score and predicted translation of current class
        dict={filename:([(it_cscore1,trans1),(it_cscore2,trans2)...],[(dialog_cscore1,trans1),(dialog_cscore2,trans2)...],[(legal_cscore1,trans1),(lega

        pat_hiddenfile=re.compile(r'^\..*')
        dict_cscore_trans={}
        for k_fname,v_sent_perplex in self.perplexs.items(): # a dict takes filename as key and a list of tuple as value, each tuple present the perple
            pattern=r'(.*)'+k_fname.rsplit('.',1)[0]+r'(.*)' # trunk of the file name :IT,Dialog or Legal
            pat_fname=re.compile(pattern)
            it_cscores_trans,dialog_cscores_trans,legal_cscores_trans=[],[],[]

            for dir in os.listdir(self.bestspec_trans_dir): #dir=Dialog, IT or Legal
                if pat_hiddenfile.search(dir):continue

                for fi in os.listdir(self.bestspec_trans_dir+'/'+dir):
                    if pat_fname.match(fi):
                        if dir=="IT":
                            it_cscores_trans=self.parsing_transfile(self.bestspec_trans_dir+'/'+dir+'/'+fi) #list of tuples (cscore_n,trans_n)
                        elif dir=="Dialog":
                            dialog_cscores_trans=self.parsing_transfile(self.bestspec_trans_dir+'/'+dir+'/'+fi)
                        elif dir=="Legal":
                            legal_cscores_trans=self.parsing_transfile(self.bestspec_trans_dir+'/'+dir+'/'+fi)
                        else:continue
            dict_cscore_trans[k_fname]=dict_cscore_trans.setdefault(k_fname, ())+(it_cscores_trans,dialog_cscores_trans,legal_cscores_trans)
        return dict_cscore_trans

    def sentence_weighting(self, cscore, perplex):
        '''this function return the weighted value for each sentence'''
        return cscore*perplex #float type

    #def set_threshold(self):
    def calcul_avr(self):
        '''this script return the perplexity avr'''
        avr_it, avr_dialog, avr_legal=0,0,0
        for perp in self.avr_perplexity:
            avr_it+=perp[0]
            avr_dialog+=perp[1]
            avr_legal+=perp[2]
        avr_it/=self.avr_perplexity.shape[0]
        avr_dialog/=self.avr_perplexity.shape[0]
        avr_legal/=self.avr_perplexity.shape[0]
        return avr_it,avr_dialog,avr_legal # tupe of macro-average perplexity of each class for all files
```

```
    def calcul_stdDev(self):
        '''return the standard deviation of perplexities for each class'''
        avr_it,avr_dialog,avr_legal=self.calcul_avr()
        stdD=np.std(self.avr_perplexity,axis=0) #[30.14258296 31.09720375 20.94262205]
        stdD_it, stdD_dialog, stdD_legal=stdD[0],stdD[1],stdD[2]
        return stdD_it, stdD_dialog, stdD_legal

    def document_weighting(self):
        dict_cscore_trans=self.get_cscore_trans()
        IT_m,Dialog_m,Legal_m=self.IT_m,self.Dialog_m,self.Legal_m #classifier
        avr_it,avr_dialog,avr_legal=self.calcul_avr()
        gener_trans=self.files2dict() #dict takes filename as key and sentence translations generated by generic model as value
        stdD_it, stdD_dialog, stdD_legal=self.calcul_stdDev()

        for k_fname,v_sent_perplex in self.perplexs.items(): #k_fname=filename, v=[(perlex_it_sent1,perlex_dialog_sent1,perlex_legal_sent1),(perlex_it_sent2,perlex_dialog_sent2
            pattern=r'(.*)'+k_fname.rsplit('.',1)[0]+r'(.*)'
            pat_fname=re.compile(pattern)
            op_file=open(output_DIR+'/'+k_fname+'.zh', 'w', encoding='utf-8')

            if self.variable==1:#document level
                pass
            if self.variable==0:#sentence_level
                for k,v in dict_cscore_trans.items(): #k=filename, v=([(it_cscores1,it_trans1),(it_cscores2,it_trans2)...],[(dialog_cscores1,it_trans1),(dialog_cscores2,dialog_
                    if pat_fname.match(k):
                        for i in range(500): # knowing that we have 500 lines in a doc

                            # #threshold=perplexity average for each class
                            #if abs(v_sent_perplex[i][0])>abs(avr_it) and abs(v_sent_perplex[i][1])>abs(avr_dialog) and abs(v_sent_perplex[i][2])>abs(avr_legal):

                            # threshold=perplexity average +- standard deviation for each class
                            if (v_sent_perplex[i][0]>avr_it+stdD_it or v_sent_perplex[i][0]<avr_it-stdD_it) and (v_sent_perplex[i][1]>avr_dialog+stdD_dialog or v_sent_perplex[i
                                for cle,valeur in gener_trans.items():
                                    if pat_fname.match(cle):
                                        op_file.write(valeur[i]+'\n')
                            else:
                                it_sent_value=self.sentence_weighting(v[0][i][0],v_sent_perplex[i][0]) #v[0][i][0] is confidence score of sentence 0, v_sent_perplex[i][0] is pe
                                dialog_sent_value=self.sentence_weighting(v[1][i][0],v_sent_perplex[i][1])
                                legal_sent_value=self.sentence_weighting(v[2][i][0],v_sent_perplex[i][2])

                                list_class=[it_sent_value,dialog_sent_value,legal_sent_value]
                                mini=min(list_class)
                                index=list_class.index(mini)
                                if index==0:
                                    op_file.write(v[0][i][1]+'\n')
                                if index==1:
                                    op_file.write(v[1][i][1]+'\n')
                                if index==2:
                                    op_file.write(v[2][i][1]+'\n')
```