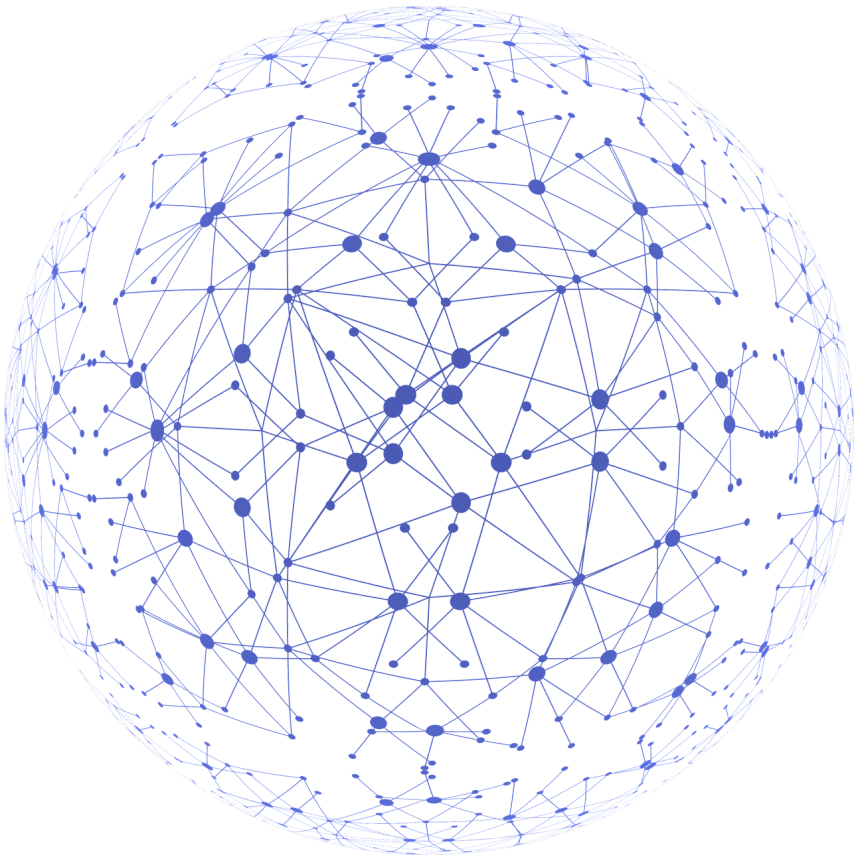
Institut national des langues et civilisations orientales
Enseignement supérieur et recherche depuis 1795



Rapport du Projet sur le moteur de traduction automatique neuronale OpenNMT

Réalisé par
THEZENAS Anissa
SAINTE LUCE Constance
NGAUV Nicolas

TRADUCTION AUTOMATIQUE ET ASSISTÉE
Dispensé par SEMMAR Nasredine
Année universitaire 2023-2024
Master 1 / Semestre 2

Sommaire

1. Introduction 2
2. Présentation du moteur de traduction neuronale OpenNMT 2
3. Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en formes fléchies..... 3
4. Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en lemmes..... 5
5. Points forts, limitations et difficultés rencontrées 6
6. Organisation..... 7
7. Annexes..... 8

1. Introduction

La traduction automatique neuronale (NMT) est une approche de traduction automatique qui utilise des réseaux de neurones artificiels pour traduire du texte d'une langue source vers une langue cible. Contrairement aux méthodes traditionnelles de traduction automatique, qui s'appuient souvent sur des règles linguistiques et des statistiques, la NMT apprend les correspondances entre les langues directement à partir de données d'entraînement.

2. Présentation du moteur de traduction neuronale OpenNMT

OpenNMT est un moteur de traduction automatique neuronale open-source développé par Harvard et SYSTRAN. Il offre une infrastructure flexible et extensible pour entraîner, évaluer et déployer des modèles de traduction automatique neuronale (NMT).

Approche utilisée :
OpenNMT est un moteur de traduction automatique neuronale qui utilise des réseaux de neurones pour apprendre à traduire du texte d'une langue source vers une langue cible. Pour ce faire, il a besoin d'un ensemble de données d'entraînement composé de paires de phrases dans les deux langues.

Le fichier de configuration toy_en_de.yaml que nous avons, définit les paramètres pour une expérience de traduction de l'anglais vers le français à l'aide d'OpenNMT. Voici une explication des différentes sections du fichier toy_en_de.yaml :

- save_data et src_vocab/tgt_vocab : ces paramètres indiquent où OpenNMT doit stocker les données d'entraînement prétraitées et les fichiers de vocabulaire.
- overwrite : ce paramètre indique si OpenNMT doit écraser les fichiers existants dans les répertoires spécifiés.
- data : cette section définit les jeux de données d'entraînement et de validation. Dans cet exemple, le jeu de données d'entraînement est composé de 10 000 paires de phrases dans les fichiers EMEA_train_10k.tok.true.clean.en et EMEA_train_10k.tok.true.clean.fr, tandis que le jeu de données de validation est composé de 1 000 paires de phrases dans les fichiers EMEA_dev_1k.tok.true.clean.enet EMEA_dev_1k.tok.true.clean.fr.
- src_vocab/tgt_vocab : ces paramètres indiquent où OpenNMT doit stocker les fichiers de vocabulaire pour la langue source et la langue cible.
- save_model : ce paramètre indique où OpenNMT doit stocker les points de contrôle du modèle pendant la formation.

- save_checkpoint_steps/train_steps/valid_steps : ces paramètres définissent la fréquence à laquelle OpenNMT doit enregistrer les points de contrôle du modèle, la durée de la formation et la fréquence à laquelle le modèle doit être évalué sur le jeu de données de validation.

Une fois que tous ces paramètres sont programmés, nous utilisons OpenNMT pour entraîner un modèle de traduction sur les données d'entraînement, puis nous utilisons ce modèle pour traduire de nouvelles phrases de la langue source vers la langue cible avec le corpus EMEA .

3. Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en formes fléchies

Corpus d'apprentissage :

- Nom : Europarl
- Description : Ce corpus contient des transcriptions de débats du Parlement européen, et traitant des sujets politiques, économiques et administratifs. Le langage utilisé dans ce corpus est donc formel.
- Nombre de phrases : 10 000
- Langue source : Anglais
- Langue cible : Français

Corpus d'évaluation :

- Nom : Emea
- Description : Ce corpus est constitué de textes en lien avec la médecine. Ils ont donc un vocabulaire spécialisé et technique en médecine. Un corpus qui peut être utile dans la traduction du domaine médical.
- Nombre de phrases : 1 000
- Langue source : Anglais
- Langue cible : Français

Métrique utilisée :
Pour évaluer les performances du modèle, nous avons utilisé le score BLEU. Le score BLEU (Bilingual Evaluation Understudy) est une métrique utilisée pour évaluer la qualité des traductions générées par des modèles de traduction automatique. Ce score doit être généralement compris entre 0 et 1. Un score BLEU plus élevé, c'est-à- dire proche de 1, indique une meilleure performance de traduction. Bien qu'en sachant cela, nous avons constaté que les résultats obtenus sont supérieurs à 1. Nous pensons par conséquent qu'il s'agit d'un pourcentage, compris entre 0 et 100 %.

Le score BLEU évalue la similarité entre une traduction qui a été générée par un système de traduction automatique, et une traduction de référence. Pour faire ce calcul, la métrique utilise des n-grammes dans une le texte généré et les compare aux n-gramme du texte traduit cible. Ces informations permettent de calculer de manière contextuelle la précision des segments des traductions de OpenNMT par rapport à ceux de la traduction de référence. Nous ne devons cependant pas oublier que ce score ne prend pas compte le sens sémantique des mots.

Pour l'évaluation, le moteur d'apprentissage de traduction neuronale OpenNMT a été réglé (tuning) sur un ensemble de 500 phrases sélectionnées à partir du corpus Europarl.

Tableau des résultats

Run	Corpus d'Évaluation	Score BLEU (%)
1	Europarl	4.300
2	Europarl + Emea	1.564

Les scores BLEU obtenus pour les deux corpus sont très faibles, avec 4.300% (annexe 1) pour le corpus Europarl et 1.564% (annexe 2) pour le corpus Emea. Ces scores indiquent une performance de très mauvaise qualité du modèle de traduction sur ces deux corpus. Des scores inférieurs à 10%, montrent que les traductions effectuées ne sont pas utilisables de par leur très mauvais rendu, car il contiendrait beaucoup trop d'erreurs. Cela serait dû au manque de correspondance entre les phrases des corpus en anglais et en français.
Pour rappel, un score BLEU de 20%-30% est considéré comme acceptable, un score de 40%-50% est dit comme très bon et un score au-dessus de 50 est exceptionnel. Dans notre cas, les scores obtenus sont très en deçà de la fourchette acceptable, avec seulement 4,300% et 1, 564%.

Il y a plusieurs raisons possibles pour lesquelles le modèle de traduction neuronale OpenNMT a obtenu de mauvais résultats sur les deux corpus évalués. Tout d'abord le manque de données d'entraînement : le modèle n'a pas été assez entraîné sur suffisamment de données pour apprendre à traduire correctement. En effet, à l'origine nous devions travailler sur des corpus de 100 000 phrases, par soucis (on va python un peu) d'appareils informatique assez performant, nous avons dû nous résigner à utiliser une quantité de phrases par corpus de 10 000 mots. Comme dit précédemment cela à engendrer des résultats non satisfaisants. Pour cette raison, nous pensons que l'ajout de plus de données d'entraînement pourrait aider à améliorer la performance du modèle.

En ce qui concerne le run numéro 2, il y a non seulement le manque de données d'entraînement mais aussi la différence de vocabulaire et de domaine de spécialisation. En effet, si le corpus d'entraînement est très différent du corpus d'évaluation, le modèle peut avoir du mal à généraliser sur le corpus d'évaluation.
Le corpus d'entraînement est utilisé pour apprendre au modèle à traduire des phrases d'une langue source à une langue cible. Le modèle est entraîné sur un grand nombre de phrases et leur traductions correspondantes, afin qu'il puisse apprendre les modèles de langues et les correspondances entre les langues. Le corpus d'évaluation quant à lui, est utilisé pour évaluer les performances du modèle sur des phrases nouvelles qui n'ont pas été vues pendant l'entraînement. Il est donc important que le corpus d'entraînement et celui d'évaluation soient suffisamment similaires. Car la différence de vocabulaire et de contexte (politique et médicale) entre les corpus d'entraînement et d'évaluation produit une baisse de la qualité de traduction, comme observé dans nos scores BLEU.

4. Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en lemmes

Description du lemmatiseur NLTK pour l'anglais :

Le lemmatiseur pour l'anglais fourni par la bibliothèque NLTK (Natural Language ToolKit) est le WordNetLemmatizer. Il permet de réduire les mots à leur forme de base ou lemme, en utilisant le corpus Wordnet.

1. Importation et Installation : Pour utiliser le WordNetLemmatizer, il est nécessaire d'importer la classe appropriée depuis NLTK. En plus, il faut télécharger plusieurs ressources de NLTK telles que les tags de parties du discours (POS tags), le corpus WordNet, et d'autres modules requis pour la tokenization des mots.
2. Initialisation : Le lemmatiseur est initialisé en créant une instance de la classe WordNetLemmatizer.
3. Fonctionnement : Le WordNetLemmatizer utilise le corpus WordNet pour effectuer la lemmatisation. La lemmatisation consiste à réduire les mots fléchis à leur forme de base, connue sous le nom de lemme. Pour améliorer la précision de la lemmatisation, le lemmatiseur utilise les tags de parties du discours (POS tags). Ces tags aident à distinguer les différentes formes grammaticales des mots, telles que les adjectifs, les noms, les verbes, et les adverbes. Cette distinction est cruciale pour choisir le lemme correct.

Pour la lemmatisation en français, nous avons utilisé le FrenchLefffLemmatizer, qui est basé sur le lexique Lefff (Lexique des Formes Fléchies du Français). Ce lemmatiseur offre les fonctionnalités suivantes :

1. Importation et Installation : Le FrenchLefffLemmatizer nécessite une installation séparée, souvent réalisée via un dépôt GitHub. Ce lemmatiseur est spécifiquement conçu pour le français et utilise un lexique riche en formes fléchies.
2. Initialisation : Le lemmatiseur est initialisé en créant une instance de la classe FrenchLefffLemmatizer.
3. Fonctionnement : Le FrenchLefffLemmatizer utilise un lexique spécifique pour le français afin de réduire les mots fléchis à leur forme de base. Il prend en compte les particularités linguistiques du français, notamment les différentes conjugaisons et formes grammaticales. Cela inclut des variations complexes dans les formes verbales, les genres et les nombres, ainsi que les accords adjectivaux.

Application aux Corpus
Nous avons développé un code pour automatiser le processus de lemmatisation sur les corpus en anglais et en français. Voici les étapes clés de ce processus :

1. Lecture des Corpus : Les corpus sont stockés dans des répertoires spécifiques, et le code parcourt ces répertoires pour lire les fichiers de texte. Les textes sont ensuite stockés dans des structures de données appropriées pour un traitement ultérieur.
2. Application de la Lemmatisation : Une fois les corpus chargés, chaque phrase du corpus est lemmatisée en utilisant le lemmatiseur approprié en fonction de la langue. Pour le français,

le FrenchLefffLemmatizer est utilisé, et pour l'anglais, le WordNetLemmatizer avec les tags POS pour une précision accrue.

3. Export des Corpus Lemmatisés : Après le processus de lemmatisation, les corpus lemmatisés sont exportés et sauvegardés dans des fichiers. Ces fichiers peuvent ensuite être utilisés pour d'autres tâches de traitement du langage naturel.

Après avoir vu la manière dont sont lemmatisé nos 2 corpus (Europarl et Emea), ont peut effectuer la manipulation pour avoir le score BLEU, comme précédemment. On peut constater qu'il y a une amélioration par rapport aux scores précédents sans lemmatisation, dans lesquels Europarl avait 4.30 % et maintenant 12.15 % ; et le second corpus avait un score de 1.56 % comparé à maintenant qui est de 7.43 %.

Tableau des résultats

Run	Corpus d'Évaluation	Score BLEU après lemmatisation (%)
1	Europarl	12.145
2	Europarl + Emea	7.433

La lemmatisation a permis de normaliser les textes, d'enlever les fléchissements du vocabulaire, ce qui a eu un impact sur les résultats de la métrique. Les résultats sont meilleurs, mais ils restent tout de même bas, ils ne font pas partie de la tranche de pourcentage acceptable, qui est nous le rappelons entre 20-30 %. Mais l'amélioration reste tout de même importante et il est bienvenu de le faire remarquer.
En revanche, pour le corpus combiné, il reste inférieur (7.43% , annexe 4) au résultat du corpus seul (12.15%, annexe 3). Ce qui reste dans la logique de l'analyse précédente, du à la différence de termes et de domaines contextuels des 2 corpus mis en situation.

Dans cette partie, bien que la lemmatisation permette au programme de faciliter la tâche de traitement, il ne faut pas minimiser la perte d'informations ; des informations qui sont apportées par les formes fléchies des mots dans le texte, qui permettent de donner un sens au texte.

Pour améliorer les résultats, comme dit pour l'évaluation de la partie précédente, nous pensons qu'en ayant 100 000 phrases au lieu de 10 000, cela aurait changé de façon significative les scores obtenus.

5. Points forts, limitations et difficultés rencontrées

Points forts :

OpenNMT est conçu de manière modulaire, ce qui permet de personnaliser facilement les différents composants du modèle selon les besoins spécifiques. Ainsi, il permet une grande flexibilité d'utilisation.
Au niveau des performances, il peut tirer parti des GPU pour accélérer l'entraînement et l'inférence

(si on en a à disposition bien sûr).
De plus, il inclut des outils pour l'évaluation des modèles, comme le calcul du score BLEU, qui aident à mesurer la qualité des traductions produites.

Limitations :

L'installation des dépendances, la configuration et l'utilisation d'OpenNMT peuvent être complexes pour des débutants (ça a été le cas pour nous).
Au niveau des ressources matérielles, OpenNMT nécessite des GPU puissants pour l'entraînement (qui peut être très long avec de grands ensembles de données) et l'inférence, ce qui peut être une limitation pour les utilisateurs disposant de ressources matérielles limitées (pour pallier cela, nous avons utilisé Google Collab et avons réduit la taille du corpus d'apprentissage pour les exercices 3 et 4, comme suggéré par le Professeur lors du dernier cours consacré à la mise en place et au démarrage du Projet).

Difficultés rencontrées :

La partie manipulation de données et prise en main nous a particulièrement donné du fil à retordre, notamment la partie sur la création des jeux de données. C'est également le cas pour le début du projet où nous devions comprendre les enjeux et créer l'environnement parfait pour commencer, ce qui était une étape cruciale pour la performance du modèle, mais cela était complexe et chronophage.

6. Organisation

Au commencement de ce projet, Anissa a réalisé le Github du groupe. Dépôt git que nous avons tous les trois mis à jour régulièrement.

Pour le code du projet, nous y avons réfléchi tous ensemble et c'est en grande partie de cette manière qu'on a réussi à avancer petit à petit et à comprendre le projet au fur et à mesure.
Chaque personne a essayé d'avancer sur une partie du code, selon les capacités des ordinateurs de chacun (pour l'entraînement par exemple), mais la réflexion s'est bien faite en groupe.
Nous nous sommes répartis les tâches comme suit :

- Nous avons tous fait l'installation (exercice 1), guidés par Anissa
- Nous avons fait l'expérimentation ensemble (exercice 2), guidés par Anissa
- Anissa s'est occupée du split du corpus
- Constance et Nicolas ont essayé l'exercice 3, mais l'ordinateur de Constance ne supportant pas la manipulation du train a dû se concentrer sur l'interprétation des résultats et Nicolas les manipulations
- Nicolas s'est occupé de l'exercice 4 (script de lemmatisation, manipulations pour vocab, train et translate, et score bleu)
- Pour la rédaction du rapport, on l'a fait à trois mais c'est en grande majorité Constance et Anissa qui ont mené la danse.

7. Annexes

Annexe 1

```
(tal-ml) ngauv-nicolas@ngauv-nicolas-ZenBook-UX463FL-UX463FL:~/Desktop/M1PluriTAL/S2/Trad_Aut
o_et_Assistee/Projet/Projet_OpenNMT/Exercice3/run1$ python3 ../../Projet_TAA/src/compute-bleu
.py ../../Projet_TAA/data/Europarl/Europarl_test_500.tok.true.clean.fr europarl_prediction.tx
t
Reference 1st sentence: il n&apos; est que juste de reconnaître la contribution extraordinair
e de ce Parlement .
MTed 1st sentence: c &apos; est pourquoi je ne pense pas que le Parlement européen .
That's 100 lines that end in a tokenized period ('.')
It looks like you forgot to detokenize your test data, which may hurt your score.
If you insist your data is detokenized, or don't care, you can suppress this message with the
 `force` parameter.
BLEU:  4.300371815610859
```

Annexe 2

```
(tal-ml) ngauv-nicolas@ngauv-nicolas-ZenBook-UX463FL-UX463FL:~/Desktop/M1PluriTAL/S2/Trad_Aut
o_et_Assistee/Projet/Projet_OpenNMT/Exercice3/run2$ python3 ../../Projet_TAA/src/compute-bleu
.py ../../Projet_TAA/data/Emea/EMEA_test_500.tok.true.clean.fr europarl_emea_prediction.txt
Reference 1st sentence: &quot; 1 .
MTed 1st sentence: il s&apos; agit d&apos; une question de l&apos; Union européenne .
That's 100 lines that end in a tokenized period ('.')
It looks like you forgot to detokenize your test data, which may hurt your score.
If you insist your data is detokenized, or don't care, you can suppress this message with the
 `force` parameter.
BLEU:  1.5641920191220855
```

Annexe 3 :

```
(tal-ml) ngauv-nicolas@ngauv-nicolas-ZenBook-UX463FL-UX463FL:~/Desktop/M1PluriTAL/S2/Trad_Aut
o_et_Assistee/Projet/Projet_OpenNMT/Projet_TAA/data/lemmatised/Exercice4_run1$ python3 ../../
../src/compute-bleu.py ../../../data/lemmatised/lemma_Europarl_test_500.tok.true.clean.fr run
1_exo4_prediction.txt
Reference 1st sentence: il n & apos ; est que juste de reconnaître la contribution extraordin
aire de ce parlement .
MTed 1st sentence: c & apos ; est pourquoi il est important d & apos ; une base d & apos ; un
 état membre
That's 100 lines that end in a tokenized period ('.')
It looks like you forgot to detokenize your test data, which may hurt your score.
If you insist your data is detokenized, or don't care, you can suppress this message with the
 `force` parameter.
BLEU:  12.14574987240099
```

Annexe 4 :

```
(tal-ml) ngauv-nicolas@ngauv-nicolas-ZenBook-UX463FL-UX463FL:~/Desktop/M1PluriTAL/S2/Trad_Aut
o_et_Assistee/Projet/Projet_OpenNMT/Projet_TAA/data/lemmatised/Exercice4_run2$ python3 ../../
../src/compute-bleu.py ../../../data/lemmatised/lemma_Europarl_test_500.tok.true.clean.fr run
2_exo4_prediction.txt
Reference 1st sentence: il n & apos ; est que juste de reconnaître la contribution extraordin
aire de ce parlement .
MTed 1st sentence: c & apos ; est la raison de l & apos ; union européen .
That's 100 lines that end in a tokenized period ('.')
It looks like you forgot to detokenize your test data, which may hurt your score.
If you insist your data is detokenized, or don't care, you can suppress this message with the
 `force` parameter.
BLEU:  7.4339397908707125
```