Camille Clavier, Laura Darenne

Projet M1 : Traduction Automatique et Assistée

I. Introduction

Problématique : analyser et évaluer le moteur de traduction neuronale OpenNMT.

II. Présentation du moteur de traduction neuronale OpenNMT

OpenNMT est une boîte à outils open-source permettant de concevoir, former et déployer des modèles de traduction neuronale automatique (NMT). Elle comprend plusieurs projets couvrant l'ensemble du processus d'apprentissage automatique, de la préparation des données à l'accélération de l'inférence (ici processus de prédiction réalisé par un modèle d'apprentissage automatique). Le travail de base d'OpenNMT comprend les étapes suivantes : préparation des données, formation du modèle, évaluation du modèle et inférence. OpenNMT prend en charge plusieurs architectures de modèle, procédures d'entraînement et tâches connexes, en privilégiant l'efficacité, la modularité et l'extensibilité[1]. Son objectif est de faciliter la recherche sur de nouvelles techniques tout en garantissant la stabilité de l'API et des performances compétitives pour une utilisation en production[1]. OpenNMT comprend une documentation pédagogique détaillée. Son approche basée sur les réseaux neuronaux apprend à traduire les phrases en minimisant la différence entre les traductions prédites et réelles dans les corpus parallèles. Une fois entraîné, OpenNMT peut traduire de nouvelles phrases en les tokenisant, en les alimentant dans le réseau neuronal entraîné, puis en sélectionnant la traduction la plus probable en fonction de la distribution de probabilité produite. Elle met l'accent sur l'efficacité, la modularité et l'extensibilité[2] pour soutenir les avancées de la recherche tout en maintenant des performances et des exigences d'entraînement raisonnables.

Pour entraîner le modèle de traduction, deux approches ont été utilisées : l'approche à base de LSTM et l'approche à base de Transformers. La première méthode n'a pas nécessité de paramètres particuliers avant de lancer l'entraînement. Ce doit être la méthode de base de OpenNMT. Par contre, pour lancer l'entraînement avec l'approche à base de Transformers, il a été nécessaire de donner un certain nombre de paramètres, obtenus sur github[3].

Les LSTM (Long Short-Term Memory) sont une architecture de réseau de neurones récurrents qui permet au modèle de capturer et de conserver les dépendances à long terme dans les séquences. Cela permet de ne pas analyser les mots un par un pendant l'entraînement mais d'analyser des phrases dans leur contexte.

Les Transformers sont une architecture de réseau de neurones différente. Ils sont beaucoup plus performants et permettent de traiter d'énormes quantités de données plus rapidement. Ils se basent notamment sur le principe de l'attention, "un mécanisme qui permet de concentrer l'attention du modèle sur les mots les plus pertinents dans une séquence de textes. Il permet au

[1] The OpenNMT Neural Machine Translation Toolkit: 2020 Edition. Guillaume Klein, François Hernandez, Vincent Nguyen, Jean Senellart. AMTA 2020. [consulté le 26/05/2022]
[2] OpenNMT: Open-Source Toolkit for Neural Machine Translation. Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, Alexander Rush. AMTA 2018. [consulté le 26/05/2022]
[3] https://github.com/ymoslem/OpenNMT-Tutorial/blob/main/2-NMT-Training.ipynb.

modèle de faire une prédiction en fonction de l'ensemble de la séquence plutôt que de simplement considérer chaque mot séparément."[4]

III. Evaluation du moteur de traduction neuronale OpenNMT sur un corpus en formes fléchies

Langue source : français
Langue cible : anglais

a. corpus Europarl

Run 1 (./data/corpus_europarl/run1_europarl)

- preprocessing : tous les fichiers ont été préalablement tokenisé et nettoyé avec les scripts moses
    - corpus train source et cible : 97 785 phrases
        - avant preprocessing : 100 000 phrases du corpus Europarl
    - corpus dev source et cible : 3 646 phrases
        - avant preprocessing : 3 750 phrases du corpus Europarl
    - corpus test Europarl : 488 phrases
        - avant preprocessing : 500 phrases
        -

A partir d'ici, tout à été fait sur google colab (./script_entrainement_run_un.ipynb).

- training : avec onmt_train
    - Steps : 100 000
    - Au 100 000 step :
        1. Train perplexity: 5.58874
        2. Train accuracy: 63.6154
        3. Validation perplexity: 11.4847
        4. Validation accuracy: 58.3787
- traduction : avec onmt_translate du français vers l'anglais
    - utilisation du fichier test du corpus Europarl
- evaluation : avec le script compute_bleu.py de MT-evaluation
    - pour le fichier test Europarl : BLEU: 29.069833672673568

Run 2 : (./data/corpus_europarl/run2_europarl)

Ce second essai a été fait avec l'aide des tutoriels proposés par @ymoslem sur github. Tout a été fait sur google colab (./run2_NMT-Data-Processing.ipynb et ./script_entrainement_run_deux.ipynb).
Il a été réalisé avec pour objectif de déterminer si le preprocessing influence le score d'évaluation et déterminer lequel des preprocessing moses ou nmt était le plus performant.

- preprocessing : Avec MT-processing - data filtering, tokenization et sub-wording sur le Europarl de 100 000 phrases.
  Je n'ai pas jugé nécessaire de faire le data-splitting étant donné que nos données étaient déjà divisées en train, test, dev sans pré-traitement.
    - corpus train source et cible : 95 047 phrases

[4] Introduction aux réseaux de neurones transformers, Ilyes Talbi. https://larevueia.fr/introduction-aux-reseaux-de-neurones-transformers/. [consulté le 26/05/2022]

    - corpus dev source et cible : 4 000 phrases
    - corpus test source et cible : 1 000 phrases

A partir d'ici, tout à été fait sur google colab (./run1_emea_europarl/run1_emea_europarl.ipynb).

- training : avec onmt_train
    - Steps : 50 000
    - Au 10 000 step :
        - Train perplexity: 8.87488
        - Train accuracy: 56.6657
        - Validation perplexity: 8.0714
        - Validation accuracy: 59.416
- traduction : avec onmt_translate du français vers l'anglais
    - utilisation du fichier test du corpus Europarl
- evaluation : avec le script compute_bleu.py de MT-evaluation
    - pour le fichier test Europarl : BLEU: 24.97663334828971

Résultats

	Run 1 : EUROPARL 100 000 steps	Run 1 : EUROPARL 50 000 steps	Run 2 : EUROPARL 50 000 steps
fichier test Europarl : 500 phrases	BLEU: 29.06983367267356	BLEU: 28.460048711253765	BLEU: 24.97663334828971

Le run 2 étant sur 50 000 steps, j'ai également effectué le score bleu sur le run 1 à 50 000 steps. Le preprocessing a donc bien une influence sur le score bleu, celui de moses semble au vu des résultats plus efficace.

b. corpus Emea - Europarl

Run 1 (./data/corpus_emea_europarl/run1_emea_europarl)

- preprocessing : tous les fichiers ont été préalablement tokenisé et nettoyé avec les scripts moses
    - corpus train source et cible : 107 666 phrases
        - avant preprocessing : 100 000 phrases du corpus Europarl et 10 000 dur corpus EMEA
    - corpus dev source et cible : 3 646 phrases
        - avant preprocessing : 3 750 phrases du corpus Europarl
    - corpus test EMEA : 500 phrases
    - corpus test Europarl : 488 phrases
        - avant preprocessing : 500 phrases

A partir d'ici, tout à été fait sur google colab (./run1_emea_europarl.ipynb).

- training : avec onmt_train, avec l'approche basée sur les LSTM
    - Steps : 100 000
    - Au 100 000 step :
        - Train perplexity: 2.89501
        - Train accuracy: 74.1129
        - Validation perplexity: 11.5629
        - Validation accuracy: 58.0998
- traduction : avec onmt_translate du français vers l'anglais

    - utilisation des fichiers test des corpus Europarl et Emea
- evaluation : avec le script compute_bleu.py de MT-evaluation
    - pour le fichier test EMEA : BLEU: 81.26402370156046
    - pour le fichier test Europarl : BLEU: 28.70271397878392

J'ai réévalué le modèle avec des fichiers détokénisés, la différence de score était de moins de 1 point. Le score bleu est assez haut pour le fichier test EMEA car il est assez court, il y a beaucoup de nombres qui ont pu être "facilement" traduit. Le score bleu du fichier test Europarl est quant à lui assez bas. Il aurait sûrement été nécessaire de faire tourner l'entraînement plus longtemps, peut-être jusqu'à 200 000 steps, mais les performances de mon ordinateur ont posé un certain nombre de limites.

Run 2 : (./data/corpus_emea_europarl/run2_emea_europarl)

Ce second essai a été fait avec l'aide des tutoriels proposés par @ymoslem sur github. Tout a été fait sur google colab (./run2_NMT-Data-Processing.ipynb et ./run2_NMT-Training.ipynb).

- preprocessing : Avec MT-processing - data filtering, tokenization, sub-wording et data splitting sur le corpus EMEA et Europarl de 100 000 phrases
    - corpus train source et cible : 95 047 phrases
    - corpus dev source et cible : 4 000 phrases
    - corpus test source et cible : 1 000 phrases
- training : avec onmt_train, avec l'approche basée sur les Transformers
    - Steps : 10 000
    - Au 10 000 step :
        - Train perplexity: 3.52576
        - Train accuracy: 99.8878
        - Validation perplexity: 2356.99
        - Validation accuracy: 18.1596
- traduction : avec onmt_translate du français vers l'anglais
    - utilisation du fichier test de 1 000 phrases obtenu avec la division du corpus 110k
- evaluation : avec le script compute_bleu.py de MT-evaluation
    - score BLEU: 0.32214757931123045

Le score est très bas et le nombre de steps aussi. Le problème rencontré pendant le train est qu'il est très long et trop gourmand pour le faire fonctionner longtemps sur mon ordinateur. Je ne peux malheureusement pas comparer les deux approches. Il est également tout à fait possible que le score bas et la longueur de l'entraînement soit dû à des erreurs de ma part.

Résultats

	Run 1 : Emea-Europarl	Run 2 : Emea-Europarl
fichier test Europarl : 500 phrases	BLEU: 28.70271397878392	N/A
fichier test EMEA : 500 phrases	BLEU: 81.26402370156046	N/A
fichier test Europarl et EMEA : 1 000 phrases	N/A	BLEU: 0.32214757931123045

Camille Clavier, Laura Darenne

IV. Evaluation du moteur de traduction neuronale OpenNMT sur un corpus en lemmes

Langue source : français
Langue cible : anglais

La lemmatisation des fichiers a été faite avec la librairie python spaCy et les modèles "en_core_web_sm" et "fr_core_web_sm". Le processus est assez simple. Le texte est d'abord tokenisé. Ensuite, chaque token est étiqueté avec des informations grammaticales. À l'aide de ces informations, spaCy applique des règles et consulte un dictionnaire pour déterminer le lemme associé à chaque mot. Chacun des modèles utilise également des règles spécifiques pour gérer les cas particuliers : formes irrégulières, constructions spécifiques, formes conjuguées des verbes et l'accord des adjectifs en français. Une fois la lemmatisation effectuée, les lemmes correspondants sont renvoyés pour chaque token.

a. corpus Europarl

Run 3 : (./data/corpus_europarl/run3_europarl_lemmatise)

Tout est identique à la première run de la partie précédente du preprocessing à l'évaluation. La seule différence est que les fichiers ont tous été lemmatisés.

- training : avec onmt_train
    - Steps : 100 000
    - Au 10 0000 step :
        - Train perplexity: 5.70493
        - Train accuracy: 62.7286
        - Validation perplexity: 9.68124
        - Validation accuracy: 59.0318

b. corpus Emea - Europarl

Run 3 : (./data/corpus_emea_europarl/run3_emea_europarl_lemmatise)

Tout est identique à la première run de la partie précédente du preprocessing à l'évaluation. La seule différence est que les fichiers ont tous été lemmatisés.

- training : avec onmt_train
    - Steps : 100 000
    - Au 10 0000 step :
        - Train perplexity: 5.78257
        - Train accuracy: 62.6157
        - Validation perplexity: 9.68466
        - Validation accuracy: 58.7848

Les scores sont un peu plus hauts car la lemmatisation du corpus enlève un certain nombre de paramètres dans la traduction : conjugaison des verbes, l'accord en nombre et en genre des adjectifs et noms…

Résultats

	Corpus Europarl	Corpus Emea-Europarl
fichier test Europarl	BLEU : 30.08176237470913	BLEU: 30.13750278906608
fichier test EMEA	N/A	BLEU: 92.11933788351024

V. Points forts, limitations et difficultés rencontrées

Au final, il aura été difficile d'évaluer correctement le moteur de traduction OpenNMT. Nos ordinateurs n'étaient pas assez performants pour tenir tout l'entraînement malgré l'utilisation de google colab.
Les scores bleus ne sont pas très hauts et cela pour plusieurs raisons possibles. Cela peut être à cause d'erreurs de notre part pendant le prétraitement des fichiers ou l'écriture des paramètres pour l'entraînement du modèle. Ce peut-être également dû à un sous-entraînement du modèle. Enfin, le corpus de train d'environ 100 000 phrases de moins de 80 caractères n'est peut-être pas assez grand pour obtenir un score bleu haut.
Cependant, OpenNMT possède une documentation claire et abondante. Les tutoriels proposés par la communauté sont super pour comprendre comment marche ce moteur de traduction. Cela nous a beaucoup aidé pour choisir les bon paramètres et lancer l'entraînement. Un autre point fort est qu'il est tout à fait possible d'entraîner un modèle de traduction avec l'approche à base de LSTM sans préciser ses paramètres, car il y en a déjà par défaut. Cela rend la prise en main du moteur de traduction plus accessible.

VI. Organisation

- Gitlab :

Nous avons décidé de créer deux gits, un premier pour le projet et un second pour le rendu. Sur le premier git, nous avons créé deux branches (une chacune). Il servait principalement à l'avancée du projet et l'organisation n'y était pas la priorité. C'est dans le second git, que nous avons suivi l'organisation du main comme indiqué dans les consignes et que nous avons juste eu à déplacer les fichiers du premier git.

- Corpus :

Laura s'est chargée du corpus emea_europarl. Camille s'est chargée du corpus europarl.

- Scripts :

On écrivait des scripts toutes les deux et après concertation on retenait le script le plus pertinent.

- Rapport

Le rapport a été écrit ensemble en partageant les informations, notamment pour discuter de nos résultats afin de définir ce que l'on pourrait mettre en avant. Lorsqu'une de nous deux écrivait une partie, l'autre relisait et faisait des commentaires desquels s'ensuivait des modifications.

Globalement nous nous sommes beaucoup appelées pour nous coordonner et avancer ensemble dans le projet.

VII. Annexes

Tous les fichiers nécessaires à la compréhension du travail fait se trouvent sur le gitlab de notre groupe : https://gitlab.com/Araule/taa-darenneclavier. Le dossier OpenNMT utilisé pendant les entraînements ne se trouve pas sur notre git. Il se trouvait à l'emplacement des fichiers utilisés (par exemple ./run1_emea_europarl) pendant l'entraînement.
Voici les dossiers :

- dossier data
    - dossier all corpora
        - tous les corpus utilisés pour le projet
    - dossier corpus_emea_europarl et corpus_europarl
        - chaque dossier correspond à une run
            - dossier models : dossier contenant le ou les modèles entraînés avec le nombre de steps
            - dossier run : le vocabulaire des textes créé et utilisé pendant l'entrainement
            - *.ipynb : les notebooks utilisés pour l'entraînement sur google colab
            - compute-bleu.py : script python utilisé pour calculer le score bleu
            - config.yaml : YAML configuration file
            - train.log : journal de l'entraînement
            - tous les fichiers textes utilisés pour l'entraînement et l'évaluation
- dossier src
    - tous les scripts python et bash utilisés pour le preprocessing des corpus
- Darenne_Clavier.pdf : le rapport de projet
- README.md : explication des scripts python et bash.