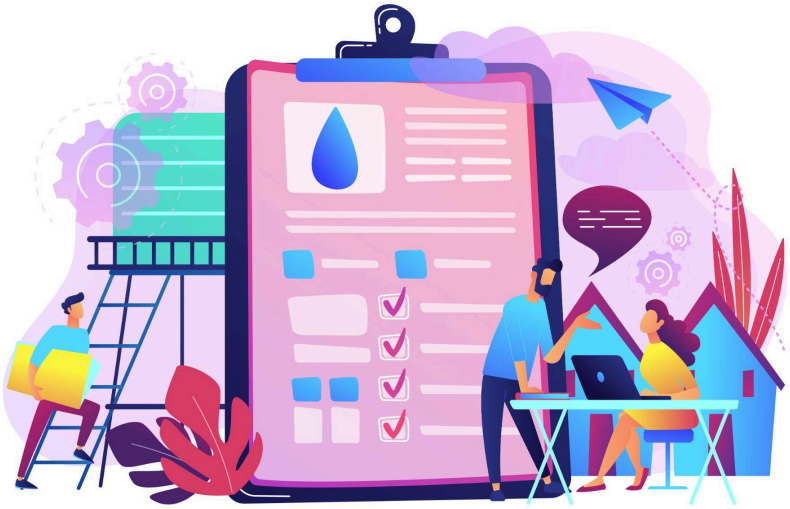
POLYTECH
PARIS-SACLAY

Traitement automatique des langues



Rapport de projet

Sujet 1 : Entraînement du modèle OpenNMT *from scratch*

Professeur : M. Nasredine SEMMAR

Auteurs : Cyril CLOVIS, Uzeir JOOMUN & Lorraine GRANDJEAN

Version 1.0

Année : 2025

Date de rédaction :

Sommaire

I/ Introduction

Dans le cadre du module de Traitement Automatique des Langues, nous devons analyser et évaluer à grande échelle un moteur de traduction neuronale : OpenNMT.

La Traduction Automatique Neuronale (TAN) est une approche de traduction qui diffère d'autres approches comme la traduction à base de règles ou encore l'approche statistique. Elle est apparue au milieu des années 2010 et a permis de réellement changer la traduction automatique faite jusqu'à présent. Par exemple, pour l'approche statistique, la traduction est faite à partir de probabilités de corrélation entre un texte en langue source et un autre en langue cible.

La TAN se base sur l'intelligence artificielle et plus particulièrement sur les réseaux de neurones pour établir des correspondances entre le texte source et sa traduction. Comme toute intelligence artificielle, il faut entraîner le modèle à traduire des textes en différentes langues. Plus le modèle est entraîné, mieux il pourra traduire les textes et surtout "comprendre" le contexte. En effet, pour plusieurs mots, il peut y avoir des sens ou traductions différentes. Dans les approches plus anciennes, les traductions pouvaient ne pas être correctes en raison des doubles sens ou du contexte par exemple.

Toute la qualité de la traduction réside également dans les données utilisées pour entraîner le modèle. On peut *fine-tuner* un modèle pour un certain type de documents à traduire. Par exemple, on peut entraîner le modèle sur des documents médicaux si l'on souhaite ne traduire que ça à l'avenir. Il va également de soi que la traduction utilisée dans les données qui serviront à l'entraînement doit être qualitative pour que le modèle le soit lors de son utilisation.

Ce rapport explique succinctement le fonctionnement de OpenNMT ainsi que les résultats obtenus. Une expérimentation sera faite avec une base de données "jouet" pour tester le bon fonctionnement du modèle. Puis des corpus issus du parlement européen (EUROPARL) et des agences médicales européennes (EMEA) seront utilisés pour entraîner le modèle dans différentes configurations afin de tester son efficacité en fonction de différents paramètres. Le but est de montrer son fonctionnement et les limites d'un tel modèle de traduction.

II/ Présentation du moteur de traduction neuronale OpenNMT

OpenNMT est un modèle de traduction automatique neuronale open-source lancé en 2016 par le groupe NLP de Harvard et SYSTRAN. Aujourd'hui, il est pris en charge par les sociétés SYSTRAN et UBIQUS.

Les principaux responsables de la maintenance sont :

- Guillaume Klein (SYSTRAN)
- François Hernandez (UBIQUS)
- Vincent Nguyen (chercheur indépendant) [1]

Il s'agit d'une approche différente de celles utilisées auparavant qui se basaient sur des approches probabilistes ou statistiques et qui ne donnaient pas forcément des résultats naturels. OpenNMT est basé sur l'apprentissage profond et utilisait à l'origine des RNN (LSTM ou GRU). Avec l'apparition des Transformers qui prennent en compte l'attention dans les textes, OpenNMT utilise maintenant cette architecture afin de rendre les traductions encore meilleures. Il priorise l'entraînement rapide et une bonne efficacité pour les tests, un maintien de la modularité du modèle et un soutien de l'extensibilité de la recherche [2]. OpenNMT utilise le modèle encodeur-décodeur que ce soit pour l'architecture LSTM ou Transformers.

L'architecture des LSTM (Long Short Term Memory) permet d'éviter que les poids soient peu modifiés lors de la rétropropagation des résultats de la fin du réseau vers le début. Il était possible de perdre des informations à cause du problème de "*Vanishing Gradient*" [3]. Des données ayant un faible gradient tendent à disparaître à mesure que de nouvelles données avec des gradients plus hauts apparaissent. Les LSTM viennent en partie combler la courte durée de vie de la mémoire des RNN. Certaines informations sont retenues par le réseau et d'autres sont oubliées de façon à ce que le modèle puisse apprendre de façon plus efficiente.

OpenNMT, plus précisément un RNN bi-directionnel, encode donc des phrases sources qui sont pré-traitées (tokenisation/lemmatisation, réduction des phrases, embedding…). Un RNN fait correspondre chaque mot source à un vecteur de mots et les traite en une séquence de vecteurs cachés [2]. L'encodeur stocke également le contexte du texte, qui est un vecteur de nombre, dans les états cachés. Chaque entrée génère son état caché. Ces derniers seront utilisés par le décodeur pour le contexte initial. Le mécanisme d'attention intervient à ce moment et permet de se focaliser sur les parties de la phrase source plutôt qu'un seul mot, ce qui améliorera ensuite la traduction.

Le décodeur reprend les états cachés générés par l'encodeur et attribue un score à chacun de ces états avec les vecteurs cachés des états afin de prédire les scores des mots suivants. Une couche softmax est utilisée pour produire une distribution des mots suivants avec la probabilité : $p(w_t \mid w_{1:t-1}, x, \theta)$ [2]. Le décodeur génère ensuite un mot à la fois en prenant en compte le mot précédent, l'état caché précédent du décodeur et le vecteur de contexte

donné par l'attention. Le vecteur de contexte et l'état caché sont concaténés et donnent ensuite la prédiction [4]. L'avantage de cette approche est qu'elle peut fonctionner avec peu de données mais prend beaucoup de temps à entraîner.

OpenNMT utilise également l'architecture Transformers qui pallie le problème de la mémoire à trop court terme pour enregistrer le contexte et qui ne repose pas sur les RNN. Transformers contient également un mécanisme d'encodeur/décodeur encore plus centré sur l'attention. Il contient un empilement d'encodeurs et de décodeurs. Chaque encodeur contient 2 sous-couches : un mécanisme de Self-attention et un feed-forward network.

Avant d'être encodée, la phrase subit également des pré-traitements (tokenisation, retrait de la ponctuation etc…) et chaque mot est transformé en un vecteur. Par ailleurs, la position du mot dans la phrase à une importance et c'est ce qui est stocké dans le vecteur. La sous-couche de Self-attention permet à l'encodeur de regarder les autres mots de la phrase et pondère leur importance. Il modifie donc le vecteur de chaque mot et le transmet au réseau Feed-Forward [5]. Ce dernier contient 2 couches de transformation linéaire et une couche d'activation ReLU entre les 2 couches et rend le résultat du vecteur passant par ces couches [6]. Le schéma de la sous-couche de self-attention et du Feed-Forward est répété autant de fois qu'il y a d'encodeurs. Entre les couches, les vecteurs sont normalisés et sommés.

Après la phase d'encodage, on peut décoder les vecteurs obtenus. Le décodeur contient une couche de Self-attention, une couche d'encodeur/décodeur attention et un réseau Feed-Forward. Globalement, le décodeur suit la même structure mais le Self-attention masque les mots après celui en cours de décodage. Le décodeur rajoute une attention supplémentaire avant de passer au réseau Feed-Forward. La dernière couche effectue une transformation linéaire et multiplie le vecteur par une softmax afin d'avoir des probabilités. La cellule avec la probabilité la plus haute est choisie comme étant le mot traduit [6]. Le processus continue jusqu'à que le token de fin de phrase soit généré.

L'avantage de Transformers est qu'il est rapide et qu'il est possible d'exécuter les opérations sur le réseau Feed-Forward en parallèle. Cependant, il a besoin de beaucoup de données et il est gourmand en mémoire. Transformers est utilisé par défaut dans OpenNMT pour ses qualités et son amélioration par rapport à LSTM.

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

a) Description du corpus d'apprentissage et d'évaluation

Premièrement, nous travaillons sur deux types de corpus:

- Europarl, ce sont des extraits de transcriptions des débats du Parlement européen.
- EMEA, contient des textes liés au domaine médical, comme des notices de médicaments. Il provient de l'agence européenne des médicaments (European Medicines Agency).

Les deux corpus, Europarl et EMEA, sont largement utilisés en traitement automatique des langues (TAL), notamment pour l'entraînement de modèles de traduction automatique. Ainsi, on travaille toujours sur 2 versions de ces corpus, la version rédigée dans une langue source (ex: anglais) et celle rédigée dans une langue cible (ex: francais).

Enfin, il est important de noter que Europarl est généraliste et institutionnel, tandis que EMEA est spécialisé et technique.

Chaque corpus se décompose en train, dev, test. L'entraînement du modèle se fait sur les corpus d'apprentissage et de développement; son évaluation sur le corpus test.

Voici la répartition des données sur lesquelles nous avons travaillé:

	Apprentissage	Développement	Test
Europarl	100k	3,750	500
Emea	10k	xxxx	500

Tableau 1 : Répartition du nombre de phrases dans les ensembles de données pour les corpus Europarl et EMEA

b) Description de la métrique d'évaluation : le score BLEU

Le score BLEU (Bilingual Evaluation Understudy) est une métrique d'évaluation automatique de la qualité des traductions. Il mesure la similarité entre une traduction générée et une ou plusieurs traductions de référence en comparant des ngrams (groupes de mots consécutifs).

Il est calculé comme suit :

1. Précision des n-grammes : proportion des n-grammes de la traduction générée qui apparaissent dans les références.
2. Pondération : BLEU combine les précisions des n-grammes de différentes tailles (généralement 1 à 4) avec une moyenne géométrique.

3. Pénalité de longueur : pénalise les traductions trop courtes par rapport aux références.

Le score varie entre 0 et 1 (souvent exprimé en pourcentage), 1 indiquant une traduction parfaitement identique à une référence.

c) Tests et tableau des résultats

Voici nos deux tests:

- Cas 1: Entraînement uniquement sur le corpus de Europarl (donc généraliste)
- Cas 2: Entraînement à partir des deux corpus Europarl et EMEA. En effet, le corpus d'apprentissage concatène les corpus d'apprentissages de Europarl et EMEA

Dans les deux cas, l'évaluation se fait sur le même domaine (corpus test de Europarl) et hors domaine (corpus test de EMEA)

	Run 1 : Europarl		Run 2 : Europarl + EMEA	
	Domaine	Hors domaine	Domaine	Hors domaine
BLEU	23.60	1.60	23.83	61.30

Tableau 2 : Scores bleu après prétraitements sur le corpus en forme fléchies

On observe une légère amélioration des résultats entre le run1 et le run2 sur le même domaine mais surtout, une très forte progression passant de 1.6% à 61.30% entre le run1 et le run2 hors domaine.

Cela s'explique par le fait qu'un modèle entraîné uniquement sur un corpus généraliste (Europarl), qui couvre des sujets variés de société, ne peut pas être performant sur un domaine technique et spécialisé comme celui des textes médicaux d'EMEA.

En revanche, en enrichissant le modèle généraliste avec un entraînement sur des données spécialisées (run2), il devient bien plus pertinent sur ce domaine. C'est une augmentation fulgurante du score BLEU, car les données de EMEA sont des médicaments, des prescriptions dans lesquelles de nombreux termes reviennent souvent. Autant de termes que le modèle a dû rencontrer lors de ses entraînements.

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

a) Le lemmatiseur NLTK pour l'anglais

Pour la lemmatisation en anglais, NLTK propose l'outil WordNetLemmatizer, qui repose sur la base de données linguistique WordNet.

La lemmatisation consiste à réduire un mot fléchi à sa forme canonique, appelée lemme, en prenant en compte son contexte grammatical. Contrairement à la racinisation (stemming), qui tronque les mots, la lemmatisation permet d'obtenir des formes lexicalement correctes.

Pour lemmatiser les corpus, 3 téléchargements de ressources sont nécessaires.

- 'wordnet' : Permet d'accéder à la base de données WordNet, essentielle pour la lemmatisation.
- 'punkt' : Nécessaire pour la tokenisation des phrases en mots.
- 'averaged_perceptron_tagger' : Permet d'assigner des catégories grammaticales aux mots pour une meilleure lemmatisation.

b) Le lemmatiseur NLTK pour le français

Pour le français, le lemmatiseur FrenchLefffLemmatizer est utilisé. Il s'appuie sur le Lexique des Formes Fléchies du Français (Lefff), un dictionnaire morphologique permettant d'associer une forme fléchie à son lemme.

Il permet une lemmatisation précise pour le français en s'appuyant sur un dictionnaire morphologique dédié, et prend en compte la nature grammaticale des mots pour éviter les erreurs de lemmatisation. Ainsi les mots "manges" deviennent "manger", et "belles", "beau". (voir les annexes pour d'autres exemple de rendu de lemmatisation (anglais et français))

Cependant, il est important de noter que cette lemmatisation est bien moins performante que la précédente, car un grand nombre de mots, notamment les verbes, ne sont pas lemmatisés.

c) Tableau des résultats

	Run 1 : Europarl		Run 2 : Europarl + EMEA	
	Domaine	Hors domaine	Domaine	Hors domaine
BLEU : sans lemme	23.60	1.60	23.83	61.30
BLEU : avec lemme	23.16	1.38	23.85	67.12

Tableau 3 : Scores bleu sans et avec lemmatiseur

Pour le Run 1, la différence entre les scores BLEU avant et après la lemmatisation est négligeable. Les résultats restent relativement similaires, sans variation significative. Autrement dit, l'utilisation ou non de la lemmatisation n'influe pas sur les performances du modèle entraîné uniquement sur le corpus Europarl.

Lors du run 2, nous observons une légère amélioration dans le domaine "Hors Domaine", le score BLEU augmente de 61.30% à 67.12%. Cette amélioration est probablement due à la redondance des termes médicaux dans les notices de médicaments, ce qui facilite la traduction après la lemmatisation, en réduisant la variabilité des phrases à traduire.

En résumé, bien que les résultats soient globalement cohérents entre les 2 runs, le run 2 montre une amélioration plus marquée dans le corpus spécialisé (Europarl + EMEA - Hors Domaine), ce qui suggère que la lemmatisation a un effet plus bénéfique pour des textes où les termes sont plus redondants et spécifiques.

V/ Points forts, limitations et difficultés rencontrées

Concernant les difficultés rencontrées, l'installation de tout l'environnement est assez fastidieuse et ne fonctionne pas sur tous les PC. En effet, Uzeir a eu des difficultés à faire fonctionner l'environnement et le test d'OpenNMT puisque ce dernier ne s'entraînait pas sur les données "toy" bien que le PC ait un GPU.

Un autre problème rencontré concerne la compatibilité des pilotes graphiques. En effet, malgré une mise à jour des pilotes, la carte graphique de certains PC ne supportait pas les versions récentes de CUDA nécessaires au bon fonctionnement de l'environnement. Cette incompatibilité a empêché l'utilisation du GPU en local, limitant ainsi nos possibilités d'entraînement et nous contraignant à chercher des solutions alternatives.

Nous avons essayé Google Colab. Mais, cette approche n'était pas pratique puisque l'on possède uniquement quelques heures d'utilisation du GPU par jour. Parmi les 3 membres du groupe, nous avions qu'un seul GPU 100% opérationnel (c'est-à-dire accessible lorsque nécessaire et pas concerné par la problématique de la compatibilité des pilotes graphiques). Cela a considérablement ralenti notre travail.

VI/ Organisation

Dans ce projet, Uzeir a réalisé la partie 1 du projet, à savoir l'expérimentation du modèle OpenNMT sur les petits corpus de 10 000 phrases du Europarl. Il a mis en place les commandes utilisées dans Google Colab pour réaliser le travail. Par ailleurs, il a également rédigé l'introduction et la présentation d'OpenNMT dans le rapport.

Cyril s'est occupé de la mise en place du dépôt github et de l'automatisation des appels aux différentes commandes via des scripts python. Il a également réalisé les tests finaux pour obtenir les scores bleus et rédigé la partie III du rapport.

La troisième partie du projet, c'est-à-dire le code de la lemmatisation et les explications partie IV du rapport, a été réalisée par Lorraine.

VII/ Annexes

a) Bibliographie

[1] *Foire aux questions*, OpenNMT, Auteur non précisé, Date non précisée, https://opennmt.net/FAQ/#, consulté le 26/02/2025

[2] *OpenNMT: Open-Source Toolkit for Neural Machine Translation*, (Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart†, Alexander M. Rush), Page 2, 2017, https://aclanthology.org/P17-4012.pdf, consulté le 2/03/2025

[3] *Qu'est-ce qu'un réseau LSTM ?*, Adib Habbou, 11/08/2022, https://larevueia.fr/quest-ce-quun-reseau-lstm/, consulté le 02/03/2025

[4] Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention), Jay Alammar, 25/05/2018, https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/, consulté le 02/03/2025

[5] The Illustrated Transformer, Jay Alammar, 27/06/2018, https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/, consulté le 02/03/2025

[6] Feed Forward Neural Network in Transformers, tutorialspoint, Auteur non précisé, Date non précisée, https://www.tutorialspoint.com/gen-ai/feed-forward-neural-network-in-transformers.htm, consulté le 02/03/2025.

b) Précisions supplémentaires

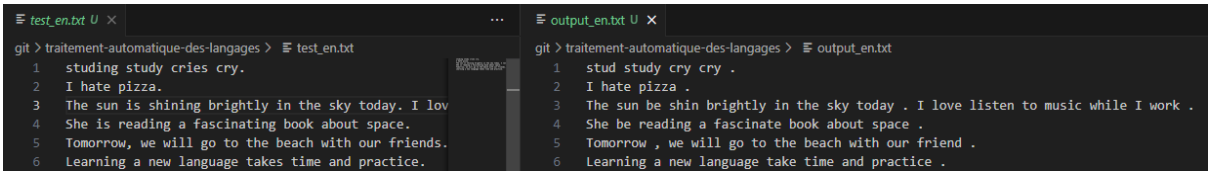


Figure 1 : Échantillon d'un texte en anglais (à gauche) lemmatiser avec nltk (à droite)

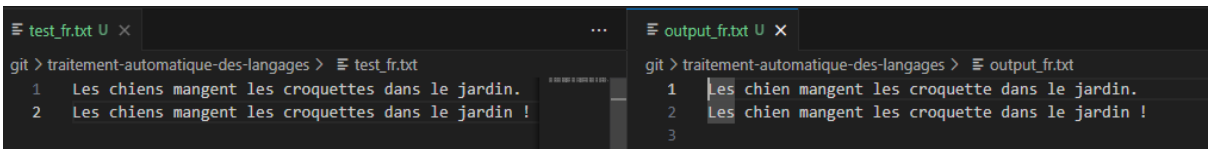


Figure 2 : Échantillon d'un texte en français (à gauche) lemmatiser avec la bibliothèque FrenchLefffLemmatizer (à droite)

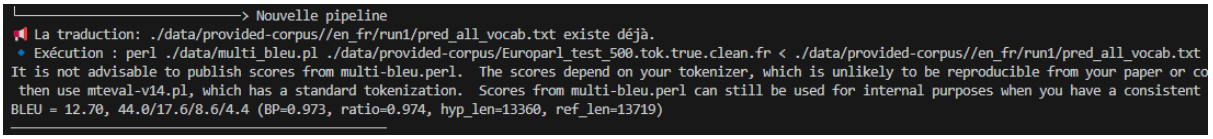
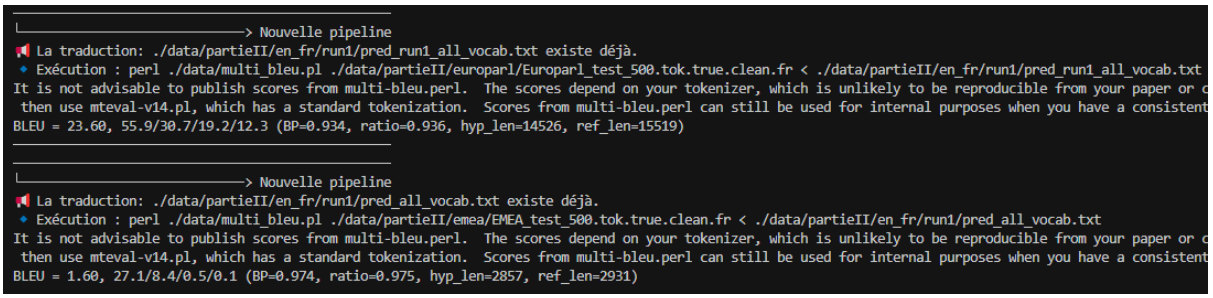


Figure 3: Score bleu pour la partie I : Mise en jambe sur le corpus fourni par le projet



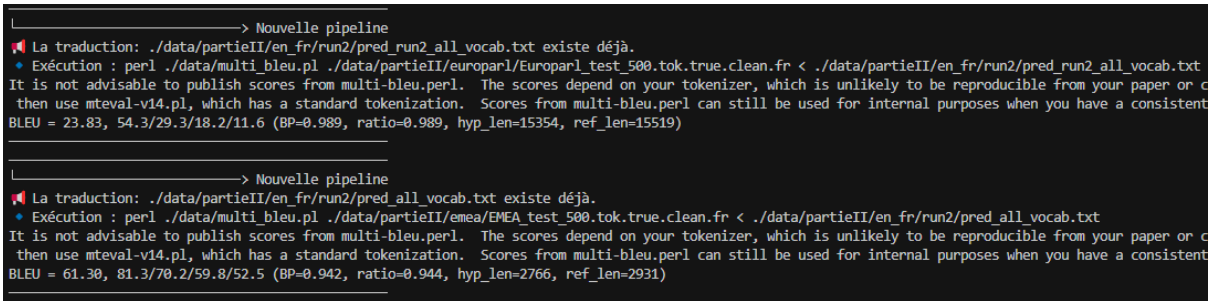
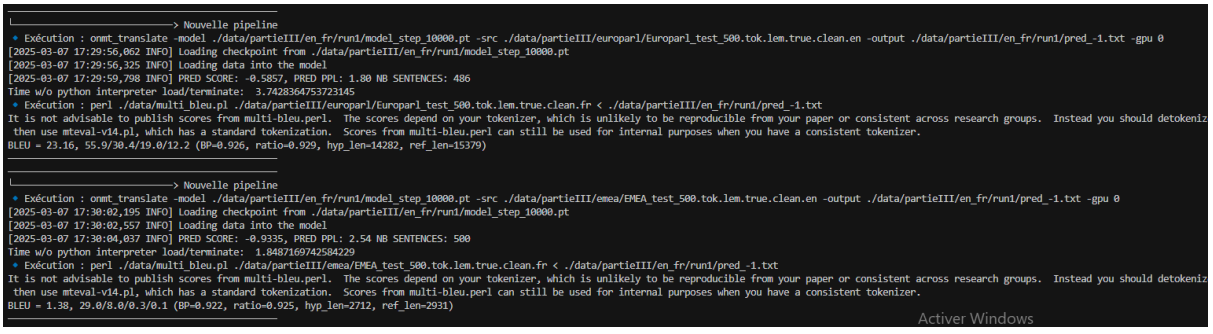


Figure 4 : Score bleu pour la partie II : Évaluation sur des corpus parallèles en formes fléchies à large échelle (run1 en haut, run2 en bas)



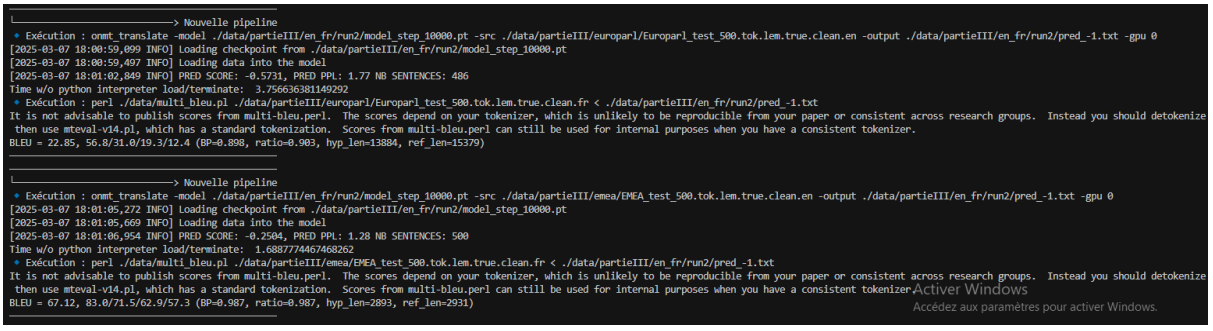


Figure 5 : Score bleu pour la partie III : Évaluation sur des corpus parallèles en lemmes à large échelle (run1 en haut, run2 en bas)

Noter bien, dans les captures données, il y a le signe " 🔹 Exécution " ou alors le signe "📢". En effet, le programme ne relance pas d'exécution s'il sait que les fichiers nécessaires sont déjà présents.

Exemple:

- python main.py II ⇒ La première fois, ca nous as pris une heure
- relance python main.py II ⇒ c'est immédiat, vous pouvez voir le score bleu.

Cela veut aussi dire que pour forcer les calculs, vous devez supprimer les fichiers correspondants.

Par ailleurs, sur notre drive, vous pouvez trouver nos modèles sur le lien ci-dessous. Vous pourrez copier-coller le dossier data et le mettre dans ./traitement-automatique-des-langages (après avoir cloné)

https://drive.google.com/drive/folders/1SzxIGCEj30czfxTmDNlDrmy9w3ZkoNOZ?usp=sharing

******** Quelques mots sur le code

C'est un ensemble de fichiers utilisant l'orienté objet et l'approche "orienté commandes". En effet, beaucoup de commandes étaient à utiliser. Les factories créent des commandes que pipeline_factory consomme !

pipeline_factory fait en quelque sorte "ses courses" chez les factories !