

**Et5-Info**

**Module :** Traitement Automatique des Langues (TAL)

**Enseignant :** Nasredine SEMMAR

**Année universitaire :** 2024-2025

**Projet Encadré**

**Consignes générales**

Ce projet correspond à une analyse et une évaluation à grande échelle d’un moteur de traduction neuronale.

Vous pouvez utiliser un dépôt ***git*** à partager entre les membres de votre groupe pour assurer la compatibilité entre les modifications que vous faites. Ce répertoire git devra être structuré comme suit :

1. Un fichier texte nommé ***README*** contenant les indications nécessaires pour exécuter les scripts Python que vous avez développés. Vous indiquerez notamment les options pour la ligne de commande ainsi que des exemples d’utilisation.
2. Un dossier ***src*** pour les codes source de vos scripts Python. La lisibilité du code sera l’un des critères de notation (commentaires précis, concis, clairs et bien orthographiés).
3. Un dossier **doc** pour votre rapport écrit en LaTeX ou en Word et compilé en un PDF de 5 à 7 pages nommé ***nom1-nom2-nom3.pdf*** qui décrit votre travail. Vous y décrirez l’objectif du projet, les résultats d’évaluation du moteur de traduction OpenNMT, ses points forts et ses limitations. Si vous savez comment résoudre ces limitations, mentionnez-le et décrivez vos idées en les présentant comme des pistes pour un travail futur. Vous devez également inclure une petite section qui décrit la contribution respective de chaque membre du groupe. Le rapport peut être écrit en anglais ou en français. Le Plan du projet est décrit dans le document « Plan\_rapport\_projet.docx ».
4. Un dossier **data** pour les données manipulées.

Vous m’enverrez par mail ([nasredine.semmar@cea.fr](mailto:nasredine.semmar@cea.fr)) uniquement un lien GitHub vers votre projet **au plus tard le vendredi 28 février 2024**.

**Travail demandé**

Vous allez analyser et évaluer le système de traduction neuronale OpenNMT. Le but de ce projet est de montrer que vous pouvez installer, évaluer et rédiger un rapport décrivant les principaux résultats, points forts et limitations d’un moteur de traduction neuronale.

**Documentation sur OpenNMT:**

* OpenNMT - Open-Source Toolkit for Neural Machine Translation: https://aclanthology.org/P17-4012.pdf
* Getting Started: https://opennmt.net/OpenNMT-py/index.html

**Installation (**30 minutes, 6 Go**)**

**Prérequis :**

Python >= 3.8

PyTorch >= 2.0 <2.2

* 1. Installer un environnement Python avec Anaconda :
     + MS Windows 10 : <https://mrmint.fr/installer-environnement-python-machine-learning-anaconda>, https://docs.anaconda.com/free/anaconda/install/windows/
     + Ubuntu >= 18.04 LTS: <https://phoenixnap.com/kb/how-to-install-anaconda-ubuntu-18-04-or-20-04>
  2. Créer et activer un environnement nommé env\_opennmt avec Python 3.9

conda create --name env\_opennmt python=3.9

conda activate env\_opennmt

* 1. Installer Gensim dans l’environnement nommé env\_opennmt

pip install --upgrade genism

* 1. Installer Numpy dans l’environnement nommé env\_opennmt

pip install numpy

* 1. Installer le framework Deep Learning PyTorch dans l’environnement nommé env\_opennmt

conda install pytorch torchvision -c pytorch

* 1. Installer OpenNMT-py dans l’environnement nommé env\_opennmt: <https://github.com/OpenNMT/OpenNMT-py?tab=readme-ov-file>
     1. En utilisant pip

pip install OpenNMT-py

* + 1. Ou bien en utilisant le code source

git clone https://github.com/OpenNMT/OpenNMT-py.git

cd OpenNMT-py

pip install -e .

pip install -r requirements.opt.txt

**Remarque :** Vous pouvez éventuellement utiliser Google Colab :

* Initiation au Deep Learning avec Google Colab (<https://moov.ai/fr/blog/deep-learning-avec-google-colab/>)
* Google Colab: charger et enregistrer des données provenant de sources externe (<https://colab.research.google.com/notebooks/io.ipynb>)
* OpenNMT Pytorch tutorial using Colab (https://colab.research.google.com/drive/1Nkd9UFlDX4NhX\_gVQwDS-77s2jV7zTqE#scrollTo=Fpgq2KISOHBo)

**Sujet 1**: Entrainer un modèle de traduction neuronale OpenNMT à partir de scratch

* + - * **How to train a model from scratch:** <https://opennmt.net/OpenNMT-py/quickstart.html>

1. **Expérimentation**
2. Vérifier l’installation d’OpenNMT sur le corpus bilingue anglais-allemand contenant 10k phrases parallèles tokenisées fourni sur le site d’installation (https://opennmt.net/OpenNMT-py/quickstart.html):

wget https://s3.amazonaws.com/opennmt-trainingdata/toy-ende.tar.gz

tar xf toy-ende.tar.gz

cd toy-ende

**Remarques :**

* Utiliser le fichier de configuration YAML « toy\_en\_de.yaml » fourni.
* La commande « gpu\_ranks: [0] » est mis en commentaire puisque la machine virtuelle n’a pas de GPU. Il faudrait enlever # si votre poste possède une GPU.
* Les instructions pour la préparation de données (Step 1: Prepare the data), l’entrainement du modèle (Step 2: Train the model) et la traduction (Step 3: Translate) sont décrites sur le lien : https://opennmt.net/OpenNMT-py/quickstart.html
* Les scripts d’évaluation de la traduction sont décrits sur le lien : https://github.com/ymoslem/MT-Evaluation. Il faut utiliser le score BLEU.

1. Vérifier que le moteur OpenNMT fonctionne correctement en utilisant les petits corpus TRAIN, DEV et TEST joints. Pour préparer les corpus pour l'apprentissage, il faut réaliser les étapes indiquées dans le document « Guide\_projet\_OpenNMT.txt »:

TRAIN : Europarl\_train\_10k.en et Europarl\_train\_10k.fr

DEV : Europarl\_dev\_1k.en et Europarl\_dev\_1k.fr

TEST : Europarl\_test\_500.en et Europarl\_test\_500.fr

1. Evaluer OpenNMT en utilisant le score BLEU sur le corpus de test.
2. **Evaluation sur des corpus parallèles en formes fléchies à large échelle**
3. Réaliser des expérimentations sur deux corpus parallèles pour le couple de langues anglais-français. La taille (en nombre de phrases) de l’ensemble des corpus bilingues utilisés pour l’apprentissage et le développement du système de traduction OpenNMT est décrite dans le tableau ci-dessous :

|  |  |  |
| --- | --- | --- |
| **N° du run** | **Apprentissage (nombre de phrases)** | **Tuning (nombre de phrases)** |
| 1 | 100K (Europarl) | 3,75K (Europarl) |
| 2 | 100K+10K (Europarl+Emea) | 3,75K (Europarl) |

1. Evaluer OpenNMT en utilisant le score BLEU. Il faudrait effectuer 2 runs avec, pour chacun d’entre eux, un test avec des données du domaine et un autre hors-domaine.

**Remarques :**

1. Pour pouvoir comparer les différents résultats, vous utiliserez les mêmes corpus d’évaluation du système de traduction statistique Moses.
2. Vous pouvez récupérer la totalité des corpus parallèles Europarl pour le domaine général et Emea pour le domaine médical à partir des liens <http://opus.nlpl.eu/Europarl.php> et <http://opus.nlpl.eu/EMEA.php>. Voir le document “Guide\_corpus\_projet” pour la recuperation des corpus Europarl et Emea à partir de la base de données Opus.

**Corpus d’apprentissage (TRAIN ) :**

* 100K (Europarl) : Europarl\_train\_100k.en, Europarl\_train\_100k.fr 🡺 Il faut prendre les 100 000 premières phrases du corpus.
* 10K (Emea) : Emea\_train\_10k.en, Emea\_train\_10k.fr 🡺 Il faut prendre les 10 000 premières phrases du corpus.

**Corpus de développement (DEV ) :**

* Europarl\_dev\_3750.en 🡺 Il faut prendre 3750 phrases à partir du corpus Europarl en commençant par la phrase au rang 100 001
* Europarl\_dev\_3750.fr 🡺 Il faut prendre 3750 phrases à partir du corpus Europarl en commençant par la phrase au rang 100 001

**Corpus de test (TEST ) :**

* Europarl : Europarl\_test\_500.en, Europarl\_test\_500.fr 🡺 Il faut prendre 500 phrases à partir du corpus Europarl en commençant par la phrase au rang 103751
* Emea : Emea\_test\_500.en, Emea\_test\_500.fr 🡺 Il faut prendre 500 phrases à partir du corpus Emea en commençant par la phrase au rang 10001

**Remarques :**

* L’idéal pour le corpus de test d’extraire de manière aléatoire un ensemble de 500 paires de phrases à partir du corpus Europarl comme un corpus correspondant au domaine et 500 autres paires de phrases à partir du corpus Emea comme un corpus hors-domaine.
* Vous pouvez utiliser la procédure train\_test\_split du module sklearn.model\_selection pour créer les corpus d’apprentissage et de test via une partition au hasard.

1. **Evaluation sur des corpus parallèles en lemmes à large échelle**
2. Lemmatiser à l’aide de la plateforme NLTK tous les corpus parallèles du domaine général (Europarl) et du domaine de spécialité (Emea) de l’exercice III.
3. Réaliser des expérimentations sur les deux corpus parallèles pour le couple de langues anglais-français selon le protocole expérimental de l’exercice III.
4. Evaluer OpenNMT en utilisant le score BLEU en utilisant le même protocole d’évaluation décrit dans l’exercice III.
5. Quelles conclusions peut-on avoir à partir de ces deux évaluations (celle de l’exercice III et celle de l’exercice IV) ?

**Remarques :**

* La lemmatisation consiste à remplacer dans les corpus parallèles du domaine général (Europarl) et du domaine de spécialité (Emea) tous les mots (en formes fléchies) par leur lemme. Les lemmes sont générés par le lemmatiseur de la plateforme NLTK. Il faut bien entendu garder la structure d’origine des corpus parallèles, c’est-à-dire une phrase par ligne.
* Pour la lemmatisation des corpus en anglais, il faut utiliser WordNetLemmatizer :

import nltk

from nltk.stem import WordNetLemmatizer

nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

**Guide d’installation et d’utilisation de WordNetLemmatizer:** https://www.holisticseo.digital/python-seo/nltk/lemmatize

* Pour la lemmatisation du français, il faut utiliser FrenchLefffLemmatizer :

from french\_lefff\_lemmatizer.french\_lefff\_lemmatizer import FrenchLefffLemmatizer

**Guide d’installation et d’utilisation de FrenchLefffLemmatizer:** <https://github.com/ClaudeCoulombe/FrenchLefffLemmatizer>

**Sujet 2**: Générer un modèle de traduction neuronale OpenNMT en utilisant un LLM pré-entrainé

* + - * **How to generate with a pretrained LLM:** <https://opennmt.net/OpenNMT-py/quickstart.html>
      * **How to finetune a pretrained LLM:** https://opennmt.net/OpenNMT-py/quickstart.html

Réaliser les trois étapes décrites dans le sujet 1:

1. Expérimentation
2. Evaluation sur des corpus parallèles en formes fléchies à large échelle
3. Evaluation sur des corpus parallèles en lemmes à large échelle