TAL : Rapport Projet n°1

25/02/23

Almeida Natanaël – Elbhar Killian – Nourry Emma

I) Introduction

La traduction automatique neuronale est une problématique complexe qui consiste à développer des modèles de traduction automatique qui sont capables de traduire des textes d'une langue à une autre de manière précise et fluide en utilisant des réseaux de neurones profonds.

La traduction automatique neuronale est un domaine de recherche en constante évolution et de nombreux défis doivent être relevés pour améliorer la qualité des traductions produites. Parmi les défis les plus importants figurent la sélection des données d'entraînement, la conception de modèles efficaces, la gestion de la complexité des langues, la prise en compte du contexte et de la sémantique, et l'évaluation des résultats.

Les modèles de traduction automatique neuronale sont souvent entraînés sur de grands corpus de textes parallèles (textes dans la langue source et leur traduction dans la langue cible). Cependant, la qualité des données d'entraînement peut avoir un impact significatif sur les performances du modèle de traduction.

Les modèles de traduction automatique neuronale doivent également être capables de prendre en compte le contexte et la sémantique pour produire des traductions précises et naturelles. En effet, les mots peuvent avoir des significations différentes en fonction du contexte, ce qui peut rendre difficile la traduction automatique.

Enfin, l'évaluation de la qualité des traductions produites est une autre problématique importante en traduction automatique neuronale. Il existe différentes métriques d'évaluation

II) Présentation du moteur de traduction neuronale OpenNMT

OpenNMT est une plateforme open source pour la construction et l'entraînement de modèles de traduction automatique neuronale. Il est conçu pour être flexible et personnalisable, offrant une grande variété de paramètres et de fonctionnalités pour répondre aux besoins des développeurs. OpenNMT est basé sur des réseaux de neurones profonds et utilise des méthodes d'apprentissage en profondeur pour améliorer les performances de la traduction automatique.

L'une des métriques d'évaluation les plus couramment utilisées pour mesurer la qualité des traductions automatiques est le score BLEU (bilingual evaluation understudy). Il s'agit d'une mesure de similarité entre la traduction automatique et la traduction de référence. Le score BLEU calcule la précision des n-grammes (séquences de n mots) dans la traduction automatique par rapport aux n-grammes de la traduction de référence.

Le score BLEU est généralement compris entre 0 et 1, où une valeur de 1 indique une traduction parfaitement précise et une valeur de 0 indique une traduction totalement inexacte. Cependant, en pratique, les scores BLEU obtenus pour les traductions automatiques sont rarement supérieurs à 0,5 ou 0,6, même pour les meilleurs modèles de traduction automatique.

OpenNMT prend en charge le calcul automatique du score BLEU pour évaluer les performances de la traduction automatique. Il fournit également d'autres métriques d'évaluation, telles que le score METEOR, le score ROUGE et le score TER, qui peuvent être utilisées pour évaluer différents aspects de la qualité de la traduction automatique.

Après avoir effectué les installations nécessaires, on vérifie que le moteur OpenNMT fonctionne correctement en suivant les instructions pour le corpus bilingue anglais-allemand :

```
(base) emmanourry@emmas-laptop-pop-os:~/Documents/ET5/TAL/TAL/Projet$
onmt_build_vocab -config toy_en_de.yaml -n_sample 10000
Corpus corpus_1's weight should be given. We default it to 1 for you.
[2023-02-22 15:15:55,469 INFO] Counter vocab from 10000 samples.
[2023-02-22 15:15:55,469 INFO] Build vocab on 10000 transformed
examples/corpus.
[2023-02-22 15:15:55,642 INFO] Counters src:24995
[2023-02-22 15:15:55,642 INFO] Counters tgt:35816
```

Entrainement :

```
onmt_train -config toy_en_de.yaml
…
[2023-02-22 15:58:41,565 INFO] valid stats calculation and sentences
rebuilding
                            took: 49.74815225601196 s.
[2023-02-22 15:58:41,565 INFO] Train perplexity: 1469.56
[2023-02-22 15:58:41,566 INFO] Train accuracy: 10.8255
[2023-02-22 15:58:41,566 INFO] Sentences processed: 64000
[2023-02-22 15:58:41,566 INFO] Average bsz: 1368/1361/64
[2023-02-22 15:58:41,566 INFO] Validation perplexity: 430.591
[2023-02-22 15:58:41,566 INFO] Validation accuracy: 13.3689
[2023-02-22 15:58:41,577 INFO] Saving checkpoint
toy-ende/run/model_step_1000.pt
```

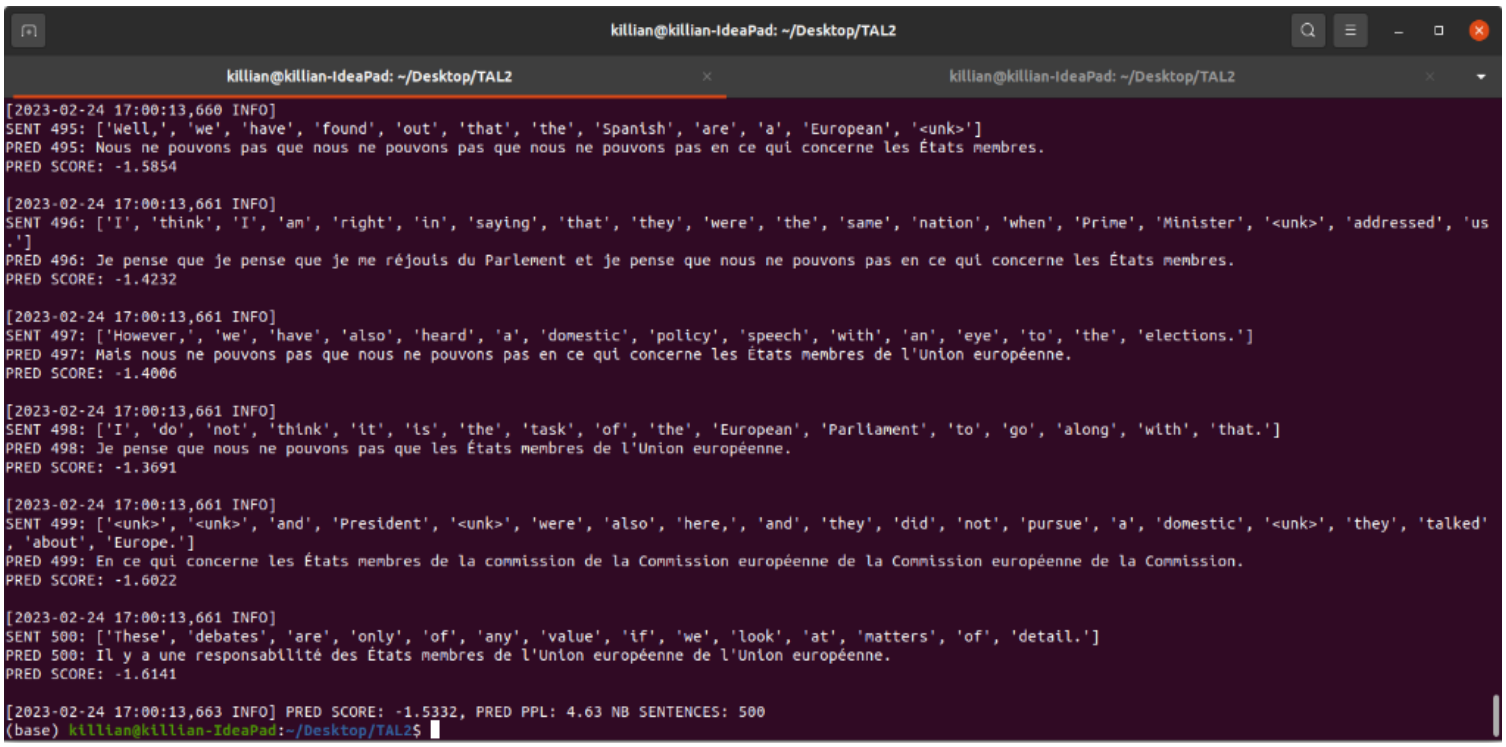
Traduction :

```
onmt_translate -model toy-ende/run/model_step_1000.pt -src
toy-ende/src-test.txt -output toy-ende/pred_1000.txt -gpu 0 -verbose

…

[2023-02-23 09:56:00,938 INFO]


SENT 2736: ['A', 'travel', 'industry', 'group', 'welcomed', 'the',
'changes', ',', 'calling', 'them', '<unk>', 'accommodations', 'for',
'a', '<unk>', 'public', 'now', '<unk>', 'with', 'technology', '.']

PRED 2736: Sie können ein , dass Sie ein , die auf der Lage und der
EU zu nutzen .

PRED SCORE: -2.5016


[2023-02-23 09:56:00,938 INFO]

SENT 2737: ['&quot;', 'We', '&apos;re', 'pleased', 'the', '<unk>',
'recognizes', 'that', 'an', 'enjoyable', 'passenger', 'experience',
'is', 'not', 'incompatible', 'with', 'safety', 'and', 'security',
',', '&quot;', 'said', 'Roger', 'Dow', ',', 'CEO', 'of', 'the',
'U.S.', 'Travel', 'Association', '.']

PRED 2737: Wir wissen , dass wir uns in der Zukunft in der Zukunft
zu nutzen , die auf das , die auf der Mitgliedstaaten zu nutzen .

PRED SCORE: -2.2212

[2023-02-23 09:56:00,941 INFO] PRED SCORE: -2.2843, PRED PPL: 9.82
NB SENTENCES: 2737
```

On vérifie également sur le corpus du TP2, pour cela on modifie le fichier toy_ende.yaml, pour qu'il prenne en compte les fichiers du corpus Europarl.

```yaml
## Where the samples will be written
save_data: run/europarl
## Where the vocab(s) will be written
src_vocab: run/europarl.vocab.src
tgt_vocab: run/europarl.vocab.tgt
# Prevent overwriting existing files in the folder
overwrite: False

# Corpus opts:
data:
    corpus_1:
        path_src: Europarl_train_10k.en
        path_tgt: Europarl_train_10k.fr
    valid:
        path_src: Europarl_dev_1k.en
        path_tgt: Europarl_dev_1k.fr


# Vocabulary files that were just created
src_vocab: run/europarl.vocab.src
tgt_vocab: run/europarl.vocab.tgt

# Train on a single GPU
world_size: 1
#gpu_ranks: [0]

# Where to save the checkpoints
save_model: run/model
save_checkpoint_steps: 500
train_steps: 1000
valid_steps: 500
```

On effectue les mêmes commandes que précédemment, pour obtenir le résultat suivant, après un temps d'exécution de 55min pour le train, on effectue le translate avec le résultat suivant, qu'on enregistre dans un fichier .txt :



Voici le résultat obtenu, suite à la mesure à l'aide du script perl pour la métrique BLEU :

```
(base) killian@killian-IdeaPad:~/Desktop/TAL2$ perl multi-bleu.perl Europarl_test_500.fr < translate_test_fr.txt
BLEU = 1.31, 14.1/2.1/0.6/0.2 (BP=0.967, ratio=0.968, hyp_len=11633, ref_len=12021)
It is not advisable to publish scores from multi-bleu.perl.  The scores depend on your tokenizer, which is unlikely to
be reproducible from your paper or consistent across research groups.  Instead you should detokenize then use mteval-v1
4.pl, which has a standard tokenization.  Scores from multi-bleu.perl can still be used for internal purposes when you
have a consistent tokenizer.
```

3) Evaluation du moteur de traduction neuronale OpenNMT sur un corpus en formes fléchies

Par manque de temps (début des stages, et temps d'exécution très élevés) nous n'avons malheureusement pas eu le temps de réaliser cette partie.

4) Evaluation du moteur de traduction neuronale OpenNMT sur un corpus en lemmes

Pour évaluer le moteur de traduction neuronale OpenNMT sur un corpus en lemmes, il est possible de lemmatiser le corpus en entrée et en sortie du modèle de traduction. Cela permet de normaliser les formes fléchies des mots en leur forme canonique (lemme), ce qui facilite la comparaison entre les traductions produites par le modèle et les traductions de référence.

Le lemmatiseur NLTK pour l'anglais utilise plusieurs règles pour la lemmatisation des mots, qui prennent en compte la catégorie grammaticale des mots et leur contexte dans une phrase. Voici quelques-unes des règles les plus courantes :

- Les noms (substantifs) sont ramenés à leur forme singulière.
- Les verbes sont ramenés à leur forme infinitive.
- Les adjectifs sont ramenés à leur forme de base.
- Les adverbes sont ramenés à leur forme de base.

Le lemmatiseur utilise également un ensemble de règles spécifiques pour les mots irréguliers en anglais. Par exemple, le mot "ran" est lemmatisé en "run", tandis que le mot "went" est lemmatisé en "go".

Le lemmatiseur utilise un dictionnaire de mots anglais avec leurs lemmes associés pour aider à identifier les formes canoniques. Ce dictionnaire est appelé WordNet et il est utilisé pour trouver les lemmes de mots qui ne suivent pas les règles de lemmatisation standard.

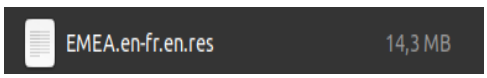
Le lemmatiseur NLTK pour le français utilise des règles similaires pour la lemmatisation des mots. Cependant, la lemmatisation en français est plus complexe que pour l'anglais, car la langue française possède de nombreuses formes verbales et nominales différentes. Pour cette raison, le lemmatiseur NLTK pour le français utilise également des règles de morphologie et de syntaxe pour identifier les formes canoniques des mots.

Tout comme le lemmatiseur pour l'anglais, le lemmatiseur NLTK pour le français utilise un dictionnaire de mots avec leurs lemmes associés pour aider à identifier les formes canoniques. Ce dictionnaire est appelé Lefff (Lexique des formes fléchies du français).

Voici le script python que nous avons écris pour la lemmatisation du corpus parallèle :

```python
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from pprint import pprint
from french_lefff_lemmatizer.french_lefff_lemmatizer import FrenchLefffLemmatizer
from nltk.corpus import wordnet

#nltk.download('omw-1.4')
#nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

def lemmatize_print(words) :
        a = []
        tokens = word_tokenize(words)
        for token in tokens:
                lemmatized_word = lemmatizer.lemmatize(token)
                a.append(lemmatized_word)
                return ((a[i] + " : " + tokens[i]) for i in range(len(a)))


with open('EMEA.en-fr.en.res', 'w') as file2:
        with open('EMEA.en-fr.en', 'r') as file:
                data = file.read().split('\n')

                for i in range(len(data)):
                        text = lemmatize_print(data[i])
                        if text is not None:
                                for j in text:
                                        file2.write(j)
```

Nous obtenons bien un résultat, cependant nous n'avons pas compris pourquoi il nous était impossible de le lire :



5) Organisation et répartition des tâches

Pour le projet, nous avons réparti les tâches comme nous le pouvions étant donné le début des stages imminents (déménagements etc.). Emma a donc travaillé sur les deux premières parties, et a essayé d'entamer la troisième. Killian a également reprit la partie deux et tenté la troisième et la quatrième. Natanaël a tenté de réaliser la troisième et la quatrième également, et a rédigé le rapport avec Killian.

6) Difficultés rencontrées

Nous avons eu beaucoup de difficultés à comprendre ce que réalisait chaque commande, et l'ordre des différentes étapes. L'analyse des résultats obtenu est également difficile pour nous. De plus, les temps d'exécution des différentes commandes (55 min pour le petit corpus) nous a semblé très contraignant, et a rendu notre progression difficile. Enfin, la période pour effectuer ce projet ne nous a pas permis d'approfondir ce dernier comme nous l'aurions souhaité (obligations personnelles liées aux débuts de nos stages).