ENSEEIHT

Apprentissage Profond
Rapport

Projet Immorthon

Élèves :
Axel Bechu
Laerian Bontinck
Clément Demazure
Vianney Hervy
Yige Yang

Enseignant :
Axel Carlier

25 mai 2025

Table des matières

1 Introduction **1**

2 Description du sujet **1**

3 Données d'entraînement **1**
3.1 Corpus . . . 2
3.2 Définitions . . . 2
3.3 Scrapping . . . 3
3.4 Préparation des données . . . 3

4 Modèle **4**
4.1 Création du modèle . . . 4
4.2 Entraînement . . . 4
4.3 Utilisation du modèle . . . 4

5 Analyse des résultats **5**
5.1 Modèle non affiné . . . 5
5.2 Modèle affiné . . . 5

6 Conclusion **5**

1 Introduction

Lorsque nous entendons pour la première fois un mot en français, il nous est parfois possible d'en comprendre le sens à l'aide du contexte ou de l'étymologie. C'est cette seconde méthode que nous allons essayer de reproduire à l'aide d'un modèle de language.

2 Description du sujet

L'objectif de notre projet tient en peu de mots : *un modèle de langage capable de générer des définitions plausibles pour des mots inventés.*

Toute la difficulté tient dans le terme "définition plausible". En effet, il faut non seulement que le texte généré soit grammaticalement correct et ait la forme d'une définition, mais il faut également que le sens donné au mot soit cohérent avec sa structure étymologique. Par exemple, le mot "étymologie" est formé des radicaux grecs ἔτυμον (vrai sens) et λόγος (discours). Il est donc logique que sa définition fasse référence à l'étude du "vrai" sens des mots. De manière générale, la plupart des mots scientifiques ont des sens assez clairs pour qui a fait des langues anciennes.

Le nom "Immorthon" est un mot-valise entre "immortel" et "python". "Immortel" est le surnom[1] donné aux académiciens de l'Académie Française, dont une partie de la responsibilité est de définir les mots de la langue française.

3 Données d'entraînement

Les données nécessaires sont sous forme de paire mot-définition. Le format JSON est parfait pour cela.

1. https://w.wiki/EDk9

Nous avions initialement prévu un corpus de mots en français. En effet, l'orthographe des mots fait souvent explicitement référence à leur étymologie (majoritairement grecque ou latine). Cependant, Il nous fallait aussi trouver un modèle français préentrainé à affiner. Il nous est apparu finalement plus facile de partir d'un modèle anglais et de l'affiner sur un corpus anglais.

3.1 Corpus

Différents corpus ont été testés : Les 3000 mots [2] les plus fréquents en anglais selon le site EF, une liste [3] de 10 000 mots utilisée par un professeur du MIT pour entraîner ses modèles et finalement un corpus de presque 500 000 mots [4].

C'est ce dernier qui a donné les meilleurs résultats. À la fois parce qu'il contient le plus de mots mais aussi parce qu'il contient de nombreux mots rares, scientifiques ou techniques. Un extrait du corpus est donné au listing 1.

Sur certains corpus, il nous est apparu que la proportion de prénoms dans notre corpus était anormalement élevée. Environ 10% des définitions étaient "a first name for boys/girls". Cela a conduit le modèle à souvent reconnaitre des mots comme des prénoms. Le corpus finalement choisi n'a que 1% de prénoms.

```
triphthong
serosynovial
syndesmoma
anton
intoxication
vajra
stue
loupcerviers
psychosis
dirtiest
```

Listing 1 – Extrait du corpus (généré avec `shuf -n 10 words-alpha.txt`)

3.2 Définitions

Une fois le corpus choisi, il nous fallait un moyen de récupérer les définitions des mots. Nous avons choisi de les extraire du site Oxford Learner's Dictionaries [5]. Ce choix est dû en partie à la qualité des définitions fournies, à la quantité de mots disponibles et surtout à la facilité de scrapper le site.

2. https://www.ef.com/wwen/english-resources/english-vocabulary/top-3000-words/
3. https://www.mit.edu/ ecprice/wordlist.10000
4. https://github.com/dwyl/english-words
5. https://www.oxfordlearnersdictionaries.com

```
"fornicators","a person who has sex with somebody they are not married to"
"spins","to turn round and round quickly; to make something do this"
"carbonization","the process of becoming or being made into carbon"
"agonies","extreme physical or mental pain"
"militarize","to send armed forces to an area"
"raptor","any bird of prey"
"biding","to stay or live in a place"
"sterility","the fact of not being able to produce children or young animals"
"armada","a large group of armed ships sailing together"
"ambiguities","the state of having more than one possible meaning"
```

Listing 2 – Extrait des définitions (généré avec `shuf -n 10 dico-alpha.csv`)

3.3 Scrapping

La difficulté principale du scrapping est de comprendre la structure du site. Il faut trouver un algorithme qui fonctionnera pour toutes les pages de définitions. On pense que ces pages sont elles-mêmes générées automatiquement à partir d'un modèle et d'une base de données de définitions. Il est donc possible de rétro-ingéniérer le modèle pour trouver comment extraire une définition d'une page.

Malgré cela, de nombreux mots du corpus choisi ont soit une page dont nous n'avons pas pu extraire la définition, soit n'ont pas de définition sur le site.

Nous nous trouvons donc avec 71010 définitions (7 fois plus que le corpus du MIT), ce qui est suffisant comme nous le verront ensuite.

Le résultat du scrapping C++ accéléré par parallélisation OpenMP [6] est au listing 3.

```
$ ./main corpora/words-alpha.txt dictionnaries/dico-alpha.csv
OpenMP is enabled.
Fetching definitions for 370105 words...
[#########----------------------------------------] 71010/370105 (19%)
71010 definitions added to dictionnaries/dico-alpha.csv
299095 definitions failed to be added.
Time taken: 18255 seconds.
```

Listing 3 – Résultat du scrapping

3.4 Préparation des données

Le modèle doit être entraîné sur des chaines de la forme `<prompt> <réponse>`. Il faut donc unifier les colonnes `word` et `definition` de nos données.

```
df["text"] = "Define: " + df["word"] + "\n" + df["definition"]
```

Listing 4 – Préparation des données

6. https://www.openmp.org

4 Modèle

La solution que nous avons choisie pour le modèle de langage est globalement très proche de celle réalisée au TP5 (Transformers pour la génération de texte).

4.1 Création du modèle

Nous avons essayé plusieurs modèles préentrainés différents (DistilGPT2 [7], GPT-2 [8]).

Le modèle est importé depuis Hugging Face [9] ou depuis une sauvegarde de notre modèle déjà affiné sur les définitions. Le tokenizer associé est également importé.

On tokenise ensuite la colonne `text` de notre base de données de définitions. Nous avons choisi un padding de 200 tokens étant donné la répartition des longueurs de définitions (figure 1).

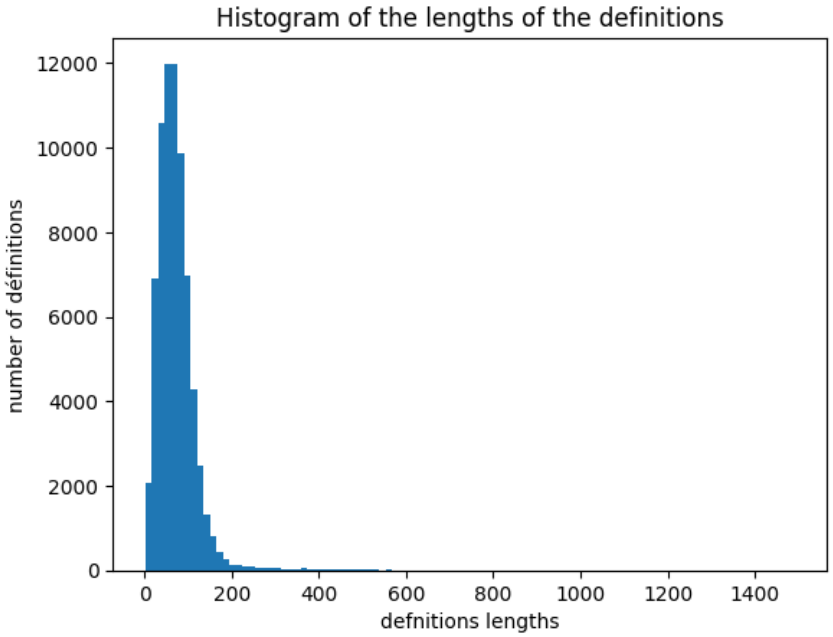

FIGURE 1 – Histogramme de la répartition des longueurs de définitions

4.2 Entraînement

L'entraînement est réalisé en une seule époque (ou plus si on réentraîne le modèle ultérieurement). La taille du batch est de 8 pour éviter la surchage mémoire des GPU.

Pour le modèle GPT-2, le temps total d'entraînement est d'environ 1h.

4.3 Utilisation du modèle

Comme lors du TP5, nous utilisons ici la fonction `pipeline` de la librairie Transformers pour générer le texte. Le modèle est chargé avec le tokenizer associé.

Le modèle étant entraîné sur des exemples de la forme `Define: <mot>\n<definition>`, il faut donc préparer le prompt d'entrée (listing 5).

7. https://huggingface.co/distilbert/distilgpt2
8. https ://huggingface.co/openai-community/gpt2
9. https://huggingface.co/

```python
def generate(prompt, numDef):
    fullPrompt = f"Define: {prompt}\n"
    results = generator(fullPrompt, num_return_sequences=numDef, ...)
    return [result["generated_text"] for result in results]

for result in generate("zoophobia", 3):
    print(result, end="\n\n")
```

Listing 5 – Fonction de génération de texte

5 Analyse des résultats

En commençant ce projet nous espérions que le modèle apprenne à reconnaitre des préfixes, suffixes et radicaux et à les associer à un sens précis. Cette appproche est parfaitement en accord avec la notion de jetons que le modèle de langage utilise pour générer du texte.

Malheureusement, le fait de patir d'un modèle préentraîné impose le tokeniser et notre affinage ne permet pas de le modifier. Il faut donc se contenter du tokeniseur du modèle de base.

5.1 Modèle non affiné

5.2 Modèle affiné

6 Conclusion