# Learning rhyming patterns with a neural language model (GPT-2)

Projet de cours ML-433, EPFL, Etienne Boisson, Bastien Bernath, Xavier Theimer-Lienhard  
Supervisé par Andrei Popescu-Belis et Àlex Atrio, HEIG-VD – novembre/décembre 2022

## Travaux préliminaires

Lectures sur le système CRPO (HEIG-VD, 2020-2022) et le système PoeLM (2022).

Prise en main du notebook N0 qui permet de générer des vers avec le modèle GPT-2 initial (à vrai dire déjà *fine-tuned* sur des poèmes) en indiquant une longueur maximale. Nécessite l’installation du package Transformers et le téléchargement du modèle GPT-2 depuis Switch Drive.

Prise en main du notebook N1 qui permet de générer automatiquement les poèmes avec CRPO mais sans passer par l’interface graphique (appel direct des fonctions de génération du backend depuis un notebook). Cela servira pour la génération de données d’entraînement et aussi de test (comparaison de la quantité de rimes).

Prise en main de la fonction qui mesure la qualité des rimes (en réalité, la similarité des *n* derniers phonèmes, selon un dictionnaire de prononciation de l’anglais) dans le notebook N2.

Étude du notebook N3 de Teo Ferrari (appelé Training-gpt2-with-english-poetry-corpus-with-start-token.ipynb) qui permet de continuer à entraîner un modèle existant avec de nouvelles données.

Estimation de la *baseline* aléatoire : générer avec CRPO des quatrains de 10 syllabes avec le modèle GPT-2 actuel, sans fixer de thèmes/émotions/RIMES, et mesurer avec N2 combien de vers riment par hasard (probablement aucun). Il faudra fixer un nombre de phonèmes (sons) qui doivent coïncider pour qu’on déclare que les vers riment : par exemple 3 ou 4.

## Expérience 1 : apprentissage du pattern AABB

Générer des quatrains de 10 syllabes avec le système CRPO, avec des rimes fixées au pattern AABB, grâce à la fonction de génération sans interface (N1). Commencer par générer 10 Mo de texte, et plus tard essayer aussi avec 100 Mo et étudier l’impact de la différence sur ce qui suit.

Joindre les poèmes (séparés par un simple '\n'), préfixer chaque vers (ligne) avec <start>, et former ainsi les données d’entraînement (en réalité, de *fine-tuning*).

Faire le *fine-tuning* de GPT-2 sur ces quatrains en utilisant le code du notebook N3.

Générer des quatrains avec le nouveau modèle : pas avec le système CRPO, mais directement via un notebook (comme N0), en limitant la longueur de la séquence et en ne gardant à chaque fois que les 4 premiers vers, pour avoir des quatrains.

Évaluer combien de quatrains ont maintenant des rimes en AABB. Est-ce mieux que la *baseline* aléatoire ?

## Expérience 2 : comme E1 mais sur un corpus de poèmes réels

Reproduire les étapes de la première expérience, mais avec un corpus avec de vraies rimes (https://github.com/sravanareddy/rhymedata) en sélectionnant uniquement dans ce corpus les quatrains rimés en AABB. Comment se comparent les scores avec E1 ?

## Expériences 3 : apprentissage du pattern ABBA, puis de ABAB (2 expériences)

Même expérience que E1, mais on vise à apprendre le pattern ABBA (séparément). Puis, on refait l’expérience avec le pattern ABAB (séparément). Comment se comparent ces deux scores avec ceux de E1 ? (On peut faire l’hypothèse que ABBA est le plus difficile à apprendre car les rimes des vers les plus éloignés doivent correspondre.)

## Expériences 4 : utilisation des modèles précédents dans CRPO (3 expériences)

Intégrer maintenant les modèles (un à la fois, p.ex. commençant par AABB) dans CRPO. Si c’est trop long, on peut ne pas intégrer le modèle de l’E2.

Donner le modèle *fine-tuned* au système CRPO, et lui faire générer des quatrains, avec des contraintes de longueur à 10 "syllabes", mais aucune contrainte explicite sur les rimes (ni sur les thèmes ou les émotions, d’ailleurs).

Est-ce que les quatrains générés riment (respectivement en AABB, puis ABBA, puis ABAB) ? Comment se compare l’usage des modèles à l’intérieur de CRPO avec l’usage indépendant (E1 et E3).

## Expérience 5 : apprentissage de plusieurs patterns dans un seul modèle

Le but est de pouvoir indiquer explicitement le pattern de rimes visé (donc utiliser un seul modèle avec plusieurs patterns possibles).

Utilise des balises ad-hoc placées au début de chaque vers, <1> et <2>, pour indiquer les vers qui riment. Les vers commenceront donc par <start> suivi d’une balise de type <1> ou <2>, avec la convention que les vers qui doivent rimer commencent par le même chiffre. (Note : les balises seront segmentées par GPT-2 en tokens <, 1, > mais cela fonctionnera quand même, comme le montre l’exemple de <start>).

Entraînement du modèle : utiliser comme corpus d’entraînement une longue séries de vers : <start><1>Un deux tr**ois**\n<start><1>Nous irons au b**ois**.\n<start><2> etc. On obtient ces fichiers en fusionnant les quatrains obtenus avec CRPO dans E1 et E3 (à partir des trois patterns AABB, ABBA, ABAB) mais en insérant avant chaque vers une balise <1> ou <2> qui indique quels vers riment entre eux.

Génération de nouveaux poèmes : utiliser CRPO et insérer en début de vers la balise <1> ou <2> selon le pattern de rimes souhaité. Par exemple, si on demande ABBA, insérer <1> puis <2> puis <2> puis <1>. Cela peut être fait aussi sans CRPO, avec quelques lignes de Python dans un notebook. Bien enlever les balises avant d’afficher le poème. Quels sont maintenant les scores de rimes pour chaque pattern ?

## Expérience 6 : un seul modèle et une balise en début de quatrain

Si le temps le permet, une autre possibilité (alternative à E5) est d’utiliser une seule balise de type <AABB> ou <ABAB> ou <ABBA> placée au début du quatrain. Elle devra donc « influencer » les 4 vers qui la suivent, ce qui semble plus difficile, mais pas impossible, d’où la nécessité de la tester empiriquement.