

Rapport de projet

Traitement de l'information

**Léo Paillé - Gatien Menaud - Alix
Renoncourt**

EI5IS102 - Traitement de l'Information
Département informatique
S5 - 2023/2024



Table des matières

1	Introduction	2
2	Présentation des données	2
3	Présentation de la méthode	2
3.1	Entraînement du modèle	2
3.2	Utilisation du modèle	3
4	Analyse du modèle et des résultats	3
5	Conclusion	3
6	Références	3

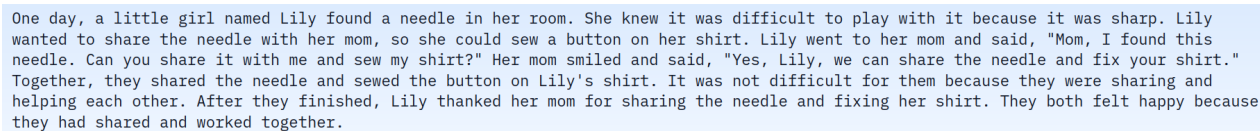
1 Introduction

Dans ce rapport, nous allons présenter les données que nous avons décidé de traiter, la méthode de traitement utilisée, puis l'analyse des résultats obtenus.

Notre objectif est de créer un modèle de génération de texte, entraîné auparavant depuis une base de données textuelles. Pour cela, nous nous appuyons en partie sur le modèle de langage GPT (transformateur génératif pré-entraîné), développé par OpenAI, ainsi que d'autres projets collaboratifs communautaires, tel que Llama2(?)

2 Présentation des données

Pour créer un modèle capable de générer du texte à partir d'un mot ou d'une phrase (appelé *prompt*), il est tout d'abord nécessaire d'entraîner ce modèle à partir de texte préexistant. Plus l'échantillon de texte est grand, plus la réponse générée pourra être complète, diversifiée et cohérente. Nous décidons pour cela de récupérer un jeu de données *TinyStories.txt*¹ (une fois décompressé), contenant plusieurs histoires de quelques lignes, en anglais, totalisant plus de 1.6G de caractères. Notez qu'une partie non négligeable de ces textes a été elle-même générée par les modèles GPT-3.5 et GPT-4. Voici un échantillon du jeu de données en figure 1 :



```
One day, a little girl named Lily found a needle in her room. She knew it was difficult to play with it because it was sharp. Lily wanted to share the needle with her mom, so she could sew a button on her shirt. Lily went to her mom and said, "Mom, I found this needle. Can you share it with me and sew my shirt?" Her mom smiled and said, "Yes, Lily, we can share the needle and fix your shirt." Together, they shared the needle and sewed the button on Lily's shirt. It was not difficult for them because they were sharing and helping each other. After they finished, Lily thanked her mom for sharing the needle and fixing her shirt. They both felt happy because they had shared and worked together.
```

FIGURE 1 – Échantillon de *TinyStories.txt*

3 Présentation de la méthode

3.1 Entraînement du modèle

À présent que nous avons un ensemble de données sur lesquelles travailler, il faut entraîner notre modèle. L'objectif étant de générer du texte lisible et cohérent, notre programme doit être capable de déterminer, à partir d'une phrase ou un mot, une suite à celui-ci. Nous devons donc lui apprendre les liens qui peuvent exister entre plusieurs groupes de mots ou de lettres, provenant d'une phrase courante depuis notre base de donnée. Dans son état final, notre modèle sera une matrice de flottants, chacun correspondant à la cohérence entre un groupe de caractères et un caractère particulier, soit la probabilité qu'un certain caractère se trouve après une suite de plusieurs autres.

Pour cela, nous allons en premier découper notre base de donnée en petites portions de texte pour ne pas surcharger l'espace de mémoire lors de l'entraînement. Ensuite, pour chacun de ces bouts de texte, nous répétons un nombre défini de fois (n_{vocab}) l'opération suivante :

- Récupérer une chaîne de caractères s à un emplacement aléatoire du texte, de longueur $context_size$

- Comparer la chaîne $s[1 : context_size - 1]$ avec le caractère $s[context_size]$ avec *gzip*
- Répéter l'opération en tronquant s par la droite jusqu'à ce que $len(s) = 2$, de manière à avoir des données pour des échantillons de taille 1 à $context_size - 1$

Les données de comparaison obtenues sont ensuite stockées dans la matrice. Par ailleurs, on conserve également toutes les chaînes s dans une liste, pour pouvoir les réutiliser lors de l'exécution du programme, et savoir à quelle chaîne de caractères correspond chaque ligne de la matrice.

Ces opérations peuvent ensuite être répétées pour chaque portion de texte, mais cela requiert beaucoup de ressources, notamment liées au matériel utilisé (CPU), et donc de beaucoup de temps. Plus le modèle est entraîné, plus il sera capable de générer du texte cohérent et diversifié.

3.2 Utilisation du modèle

Une fois que l'on a obtenu notre matrice correspondant à notre modèle, il est simple et rapide de faire générer du texte par un programme prenant en entrée un prompt écrit par l'utilisateur, ainsi que notre modèle, et générant en sortie du texte. Ce texte se développe alors caractère par caractère de la manière suivante :

- Le programme récupère les $max(taille_prompt, context_size)$ derniers caractères du prompt
- Il regarde dans le modèle à quelle ligne correspond la chaîne de caractère ainsi étudiée
- Il choisit le caractère pour lequel la valeur dans la matrice est la plus grande (ou bien un caractère aléatoirement parmi tous ceux pour lesquels la valeur est supérieure à un nombre prédéfini)
- Il affiche ce caractère, puis réitère l'opération jusqu'à une condition d'arrêt (fin de phrase, nombre de caractères requis par l'utilisateur atteint, arrêt forcé...)

4 Analyse du modèle et des résultats

Le modèle que nous obtenons est satisfaisant, à partir d'un certain nombre d'entraînements. En effet, si le modèle n'est entraîné que sur une partie restreinte de texte, le programme risque de ne pas avoir assez de données pour générer du texte, et donc de se répéter, voire de générer du texte illisible.

5 Conclusion

6 Références