

# HomerBot : Chatbot conversationnel imitant la personnalité de Homer Simpson

Jesse Dingley   Victor Maillot   Baptiste Urgell  
{dingleyjes, maillotvic, urgellbapt}@cy-tech.fr  
CY Tech

## 1 Introduction

Homer Simpson est un personnage iconique de la série télévisée d'animation mondialement apprécié : *Les Simpson*. Homer est le père de la famille Simpson, une famille typique américaine de classe ouvrière. Homer est peut-être paresseux, maladroit et alcoolique, mais il est toujours plein d'humour, dévoué à sa famille et possède un bon cœur.

Notre travail s'intéresse à essayer de donner vie à ce personnage au travers d'un chatbot conversationnel. Nous allons donc entraîner un chatbot imitant la personnalité de Homer grâce au modèle conversationnel DialoGPT (Zhang et al., 2019) et au script des *Simpson*. Nous allons tout d'abord présenter les données et leur mise en forme, puis nous présenterons en détail le modèle ainsi que sa phase d'apprentissage. Finalement, nous terminerons avec l'évaluation du modèle, l'interface utilisateur puis une ouverture.

## 2 Méthode

### 2.1 Jeu de données

**Origine des données** Afin de créer un chatbot imitant la personnalité de Homer Simpson, il nous faut beaucoup de données de dialogues du père le plus emblématique de la télévision américaine. L'ancien site du groupe *FX Networks* regroupant l'ensemble des épisodes des *Simpsons* appelé *Simpsons World*<sup>1</sup>, proposait du contenu exclusif en plus des épisodes (avant la migration de la série sur *Disney +* en novembre 2019). On pouvait notamment y trouver les scripts de la série, c'est grâce à cela que Todd W. Schneider<sup>2</sup> regroupa la totalité des dialogues des 27 premières saisons (correspondant à 596 épisodes) afin de créer une base de données pour réaliser une analyse syntaxique sur le sujet<sup>3</sup>.

<sup>1</sup><http://www.simpsonsworld.com>

<sup>2</sup><https://toddschneider.com>

<sup>3</sup><https://toddschneider.com/posts/the-simpsons-by-the-data>

**Présentation** Le jeu de données que nous allons utiliser (se basant sur le travail de Todd W. Schneider) se trouve sur *Kaggle*<sup>4</sup> et regroupe simplement l'ensemble des dialogues des 27 premières saisons des *Simpsons* en rattachant chaque phrase à son personnage.

### 2.2 Preprocessing

**Nettoyage** Le fichier au format .csv contient 158 314 lignes mais il est à noter que chaque scène est séparée par une ligne vide biaisant le nombre total de répliques. En plus de cela, certaines lignes sont incomplètes (manque le nom du personnage ou réplique vide) et doivent donc être supprimées. Finalement, une fois que les lignes vides séparant les scènes et les lignes incomplètes sont supprimées du jeu de données nous obtenons 131 853 véritables répliques.

**Data exploration** Observons maintenant la répartition du nombre de répliques par personnage. Par exemple, pour la famille Simpson (Homer, Marge, Lisa et Bart) leur nombre de répliques se répartit ainsi :

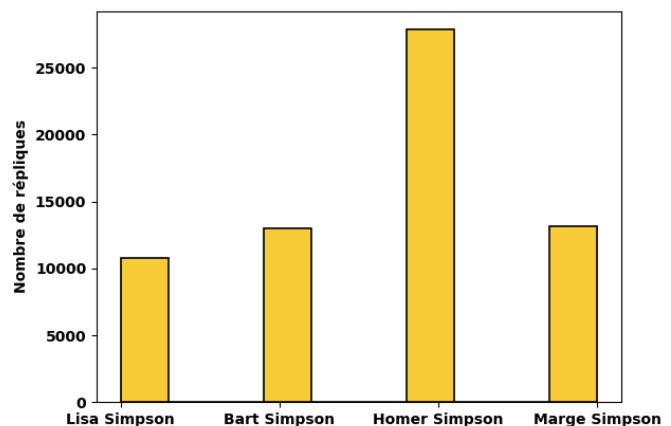


Figure 1: Nombre de répliques prononcées par la famille Simpson.

<sup>4</sup><https://www.kaggle.com/pierremegret/dialogue-lines-of-the-simpsons>

Nous pouvons remarquer que Homer avec ses 27 850 répliques domine le cercle familial, il représente à lui seul 21% du show. De plus, le nombre cumulé de répliques provenant de ces 4 personnages (64 773) représente tout de même 49% du nombre total de répliques. Nous disposons alors d'assez de données pour entraîner HomerBot.

Intéressons-nous maintenant aux répliques prononcées par Homer. Le graphique suivant illustre la longueur de ses répliques.

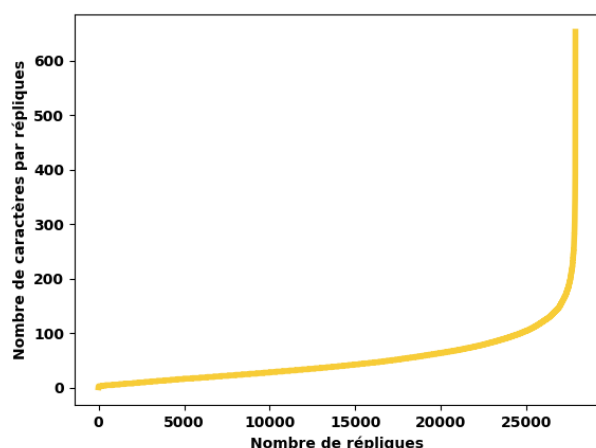


Figure 2: Longueurs des répliques de Homer classées par ordre croissant.

Nous constatons que la grande majorité (26 920 soit 96%) des répliques de Homer contiennent entre 0 et 150 caractères. Afin d'éviter que HomerBot génère de trop longues réponses, nous décidons de seulement garder les répliques de longueurs strictement inférieures à 150 caractères. D'un point de vue inverse afin d'éviter des réponses trop courtes, nous gardons alors les répliques d'Homer de longueur comprise entre 20 et 150 caractères. Cette sélection des données a pour but de normaliser les répliques et d'éviter de prendre en compte des valeurs excentriques. Après ce tri, nous possédons 20 173 répliques du personnage principal de la série (soit 72% du total de ses interventions). Nous appliquons le même processus sur l'ensemble du jeu de données afin de généraliser la normalisation. Dans la prochaine partie, nous allons voir comment mettre ses données en forme pour l'entraînement.

**Mise en forme** Le but de notre modèle sera de prédire la réponse de Homer en fonction des répliques précédentes. Nous allons sauvegarder le preprocessing dans un fichier .csv tel que chaque

ligne contienne un dialogue (succession de répliques que l'on appellera le "contexte") se terminant par une intervention d'Homer (que l'on appellera la "réponse"). L'idée générale est que le modèle s'entraînera à partir du contexte pour générer une réponse du point de vue de Homer (cf 2.3). La taille du contexte est fixe et contient généralement entre 5 et 7 répliques ((Zhang et al., 2019)). Observons comment ce mécanisme avec un contexte de taille 5 transforme le dialogue suivant :

*Marge Simpson:* So, did you kids enjoy the candy?

*Lisa Simpson:* Yeah, Mom!

*Marge Simpson:* Great.

*Homer Simpson:* But you didn't give me any!

*Bart Simpson:* Haha loser!

*Homer Simpson:* Why you little-

Le résultat en figure 3 présente donc 2 lignes correspondant au nombre de répliques de Homer au sein du dialogue. A noter que seuls les contextes issus de la même scène que la réponse de Homer sont pris en compte afin d'accentuer la cohérence des échanges. C'est pour cela que la première ligne de la figure 3 contient deux contextes vides.

Une fois avoir mis en forme chacune des répliques d'Homer avec un contexte de taille 5, nous nous sommes rendus compte que laisser des contextes vides n'étaient pas bénéfiques pour l'entraînement du modèle en raison de la forte dispersion du nombre de caractères en contextes. Nous décidons alors de supprimer les lignes du fichier .csv contenant au minimum 1 contexte vide. Après avoir entraîné HomerBot sur ces données (fine-tuning détaillé dans la partie 2.3), nous avons observé des résultats peu satisfaisants (répétitions, réponses incohérentes et éloignées de la personnalité d'Homer). C'est ce qui nous a amené à modifier la taille du contexte. Notre première idée a été d'augmenter la taille du contexte à 7 répliques mais les résultats furent tout aussi décevants. Puis nous avons eu l'idée d'entraîner le modèle seulement sur les répliques d'Homer avec donc un contexte de taille 0 afin de forcer la personnalité, mais en conséquence la discussion avec HomerBot était difficile car il n'était pas habitué à échanger. Grâce à Nguyen and Morales (2017) qui obtenait de bons résultats (pour un chatbot imitant des personnages célèbres de la télévision américaine) avec un contexte de taille 1, nous avons donc décidé de faire la même chose.

	Contexte5	Contexte4	Contexte3	Contexte2	Contexte1	Reponse
0			So, did you kids enjoy the candy?	Yeah, Mom!	Great. But you didn't give me any!	
1	So, did you kids enjoy the candy?	Yeah, Mom!	Great. But you didn't give me any!	Haha loser!		Why you little!!!

Figure 3: Exemple de mise en forme d'un dialogue avec un contexte de 5 répliques.

Cet ajustement du nombre de contexte permettra de cerner plus facilement la personnalité d'Homer. En effet, le modèle (présenté dans la partie 2.3) cherche à prédire le mot suivant à partir des précédents. En limitant la taille du contexte, le modèle s'entraînera plus souvent à prédire directement les répliques d'Homer après celle d'un autre personnage et donc à s'imprégner plus facilement de la personnalité du personnage principal des *Simpsons*.

Néanmoins, il restait un problème. En sachant que le contexte comporte seulement une réplique, il se peut qu'en plus de la réponse, l'unique réplique provienne également d'Homer. Ces répétitions présentent dans le jeu de données s'expliquent lorsque les personnages marquent une pause dans un monologue ou par effet comique. Afin d'éviter ces possibles répétitions car nous voulons entraîner le modèle sur la génération de réponses d'Homer à partir d'une autre personne que lui. Nous avons décidé de concaténer les répliques de tous les personnages, par exemple ce dialogue :

**Marge Simpson:** Get back in bed.  
**Homer Simpson:** No.  
**Homer Simpson:** All right.  
**Homer Simpson:** But I can't sleep. Can I take a walk?  
**Marge Simpson:** Sure.  
**Marge Simpson:** But Homer, don't go to Moes!

Devient après concaténation :

**Marge Simpson:** Get back in bed.  
**Homer Simpson:** No. All right. But I can't sleep. Can I take a walk?  
**Marge Simpson:** Sure. But Homer, don't go to Moes!

En faisant cela et en ayant un contexte de taille égale à 1, il sera alors impossible que la réplique précédant la réponse d'Homer soit de lui également. Comme expliqué précédemment, il sera alors plus simple pour le modèle de s'imprégner de la personnalité du père de Bart, Lisa et Maggie. Finalement, la sortie du preprocessing renvoie un fichier .csv de 2 colonnes : la première contenant le contexte de

taille 1 (soit la réplique précédant la réponse), la seconde étant la réponse d'Homer. Le fichier final *output\_preprocessing\_homer\_1\_concat.csv* contient 10 773 répliques du personnage principal des *Simpsons*.

## 2.3 Modèle et fine-tuning

**DialoGPT** Pour créer HomerBot, nous nous basons sur le modèle DialoGPT (Zhang et al., 2019) de Microsoft. Ceci est un modèle neuronal de génération de réponses conversationnelles. Plus précisément, DialoGPT est formulé en tant que modèle de langage autorégressif comme GPT2 (Radford et al., 2019) de OpenAI, et hérite de l'architecture Transformer multi-couche (Vaswani et al., 2017) de GPT2. Pour rappel, un modèle de langage autorégressif prédit le prochain mot dans une séquence en fonction des précédents mots dans cette séquence. Cependant, contrairement à GPT2 qui est un modèle générique, DialoGPT est entraîné sur 147 millions d'instances de dialogues extraites de conversations Reddit datant de 2005 à 2017. Grâce à cela, DialoGPT est capable d'adopter un comportement conversationnel et peut atteindre une qualité de réponse quasi humaine. A noter également que DialoGPT existe en trois versions de tailles différentes: DialoGPT<sub>small</sub> (12 couches et 117M de paramètres), DialoGPT<sub>medium</sub> (24 couches et 345M de paramètres) et DialoGPT<sub>large</sub> (48 couches et 762M de paramètres). Par la suite nous nous intéressons à DialoGPT<sub>small</sub> et DialoGPT<sub>medium</sub>.

Afin d'approfondir, détaillons le modèle de langage de DialoGPT. En considérant un dialogue (une succession de répliques), on commence par concaténer les répliques en un long texte  $x_1, \dots, x_N$  où  $N$  est la longueur de la séquence. Nous cassons ensuite cette séquence selon un indice  $m$ . Nous définissons tous les mots précédant  $m$  comme la source  $S = x_1, \dots, x_m$  (l'historique du dialogue) et tous les mots après  $m$  comme la cible  $T = x_{m+1}, \dots, x_N$  (réponse réelle). Le but devient ainsi d'optimiser l'objectif  $P(T|S)$ : la probabilité de la phrase cible sachant la phrase source  $S$ .

## Tokenization et création d'un Dataset PyTorch

Il est également important de parler de la tokenization des données et de la création d'un Dataset PyTorch<sup>5</sup>. Ce sont des étapes cruciales qui rendent compatibles les données avec le modèle d'entraînement. Pour rappel, la tokenization convertit les mots d'un texte en leurs identifiants d'entiers naturels respectifs ("hornet" → 6942 par exemple).

En sortie du preprocessing, nous avons un dataframe où chaque ligne correspond à un dialogue et les colonnes correspondent aux contextes et à la réponse de Homer. Par exemple si nous avons une configuration d'un contexte de taille 2:

	Contexte2	Contexte1	Reponse
0	I love you!	Do you?	Yes I do

Figure 4: Exemple d'une ligne en sortie du preprocessing à un contexte de taille 2.

Le modèle demande un Dataset PyTorch qui est essentiellement une liste de dialogues tokenisés. Pour ce faire, pour chaque ligne dans le dataframe, nous concaténons l'ensemble des cellules pour former un seul long texte  $x_1, \dots, x_N$  comme expliqué précédemment. Par exemple l'extrait de dialogue de la figure 4 sera transformé en "I love you! Do you? Yes I do". En sortie nous obtenons donc une liste de string où chaque string est un dialogue entier.

Le tokenizer que l'on utilise est le tokenizer de GPT2 qui est basé sur un BPE (Byte-Pair-Encoding) qui tokenize au niveau morphologique.

On tokenize chaque string (un dialogue entier) avec un padding de 768 (dimension de GPT2) et avec troncature. C'est à dire qu'on concatène des tokens de padding "<|padl>" aux dialogues plus courts que 768 tokens et on tronque les dialogues plus longs que 768 tokens. Ainsi le résultat de la création du Dataset est une matrice de dimension (Nombre de dialogues × 768).

**Remarque** En conséquence du format du Dataset Pytorch (où chaque dialogue est transformé en un seul string), le modèle ne va donc pas se baser sur le contexte pour prédire la réponse de Homer, mais le modèle va plutôt suivre un processus itératif où le but sera de prédire le deuxième mot du dialogue en fonction du premier, puis de prédire le troisième mot en fonction des deux premiers, etc. Cela est en partie à cause de l'implémentation avec Hugging Face (Wolf et al., 2020) qui force cette méthode causale.

**Fine-tune DialoGPT** Pour revenir à notre système HomerBot, comme nous l'avons dit, on se base sur DialoGPT. Nous fine-tunons ce modèle sur les scripts des *Simpsons* (les détails sont explicités dans les parties précédentes). DialoGPT pré-entraîné nous fournit déjà un chatbot générique capable de s'engager dans une conversation. Il nous manque à rajouter la personnalité de Homer Simpson.

Lors d'un fine-tune classique, il est standard de poursuivre avec un entraînement des dernières couches du réseau et de geler les premières couches : la raison étant que les premières couches encodent généralement la forme de l'entrée (la syntaxe et la morphologie) et les dernières couches le fond (la sémantique). Dans un premier temps nous avons procédé avec un fine-tune en gelant les premières couches du réseau (deux tiers des couches) mais cela nous avait produit des résultats décevants : des répétitions continues de mots, des incohérences (réponses hors sujet) et la génération de la même réponse à plusieurs reprises. Nous n'avons pas pu expliquer les raisons du dérèglement du modèle dans le cas d'un gel de certaines couches. Finalement, nous avons décidé de procéder avec un fine-tune sur l'entièreté des couches de DialoGPT. Cette méthode nous donne des résultats beaucoup plus cohérents.

Pour fine-tuner DialoGPT sur l'intégralité de ses couches, nous avons testé plusieurs approches. Premièrement, nous étions partis du modèle allégé "small" de 12 couches que nous avons entraîné pour 3 epochs. Cette configuration était source d'erreurs d'incohérence (HomerBot répondait hors sujet) et parfois des répétitions. Mais le bot pouvait quand même s'exprimer correctement, était compréhensible et avait la personnalité de Homer. Exemple :

```
>>> User: Where is Springfield?
HomerBot: ... What's this place called?
>>> User: Springfield.
HomerBot: ... Springfield, Michigan?
>>> User: So Springfield is in the state of Michigan?
HomerBot: I just don't know what to think..
>>> User: Are you dumb?
HomerBot: I'm not a genius.
>>> User: I'm sorry...
HomerBot: ... I know.
```

Donc ici, nous pouvons noter par exemple la répétition de "..." et nous pouvons relever que HomerBot répond un peu à côté à sa première réplique. Mais dans l'ensemble ce n'est pas terrible. A noter qu'il y a des conversations réalisées avec HomerBot à ce stade qui présentaient plus d'incohérences que l'exemple montré au-dessus.

<sup>5</sup><https://pytorch.org/docs/stable/data.html>



Dans un second temps, afin de pousser l'efficacité de notre modèle, nous avons procédé avec un fine-tune (de même sur 3 epochs) qui implémente DialoGPT<sub>medium</sub>. Passer par cette méthode a aidé à réduire drastiquement les erreurs d'incohérences mais ce modèle *medium* étant un modèle deux fois plus grand que le modèle *small* (passage de 12 à 24 couches), la personnalité de Homer dans ce cas est légèrement diluée. Parfois nous avons l'impression de parler seulement avec un DialoGPT générique. A noter également que la perplexité de ce modèle est de 36.18. Pour rappel, une perplexité de 36.18 signifie que HomerBot hésite en moyenne entre 36.18 mots à chaque étape de génération (détails de la génération sont expliquées en **3.4 Génération de réponses**). La perplexité définit le niveau d'indécision du modèle. Ci-dessous un exemple de dialogue avec la configuration DialoGPT<sub>medium</sub> + 3 epochs :

```
>>> User: Beer or TV?
HomerBot: ... or both?
>>> User: You need to go on a diet Homer...
HomerBot: ... and lose weight.
>>> User: But first, do you want to go to
        Moe's or the candy store later?
HomerBot: The candy store of course.
>>> User: What is your name?
HomerBot: My name is Homer Simpson.
>>> User: Ok. Bye see you later!
HomerBot: I'll be right over.
```

Donc nous pouvons voir qu'utiliser la configuration DialoGPT<sub>medium</sub> et 3 epochs nous donne des résultats intéressants. Mais comme dit précédemment, la personnalité de Homer est plus difficile à faire ressortir avec ce plus grand modèle donc nous avons décidé d'essayer de continuer à entraîner HomerBot<sub>medium</sub> pour 2 epochs supplémentaires. L'évolution des loss d'entraînement et de validation (sur les 5 epochs) est présentée dans la figure 5. Comme nous pouvons observer, le modèle subit un sur-apprentissage, surtout à partir du troisième epoch (6k steps) : avant le troisième epoch, le loss de validation est assez stable et le loss d'entraînement diminue progressivement. Cependant, lorsqu'on continue l'entraînement à partir de 3 epochs, nous constatons une croissance plus rapide du loss de validation et une diminution progressive du loss d'entraînement. Le résultat de ce sur-apprentissage est illustré par la perplexité du modèle (58.59). Cette valeur est 62% plus grand que la perplexité du modèle à 3 epochs. Cette version est ainsi beaucoup plus indécise. Nous conservons donc la version HomerBot<sub>medium</sub> à 3 epochs.

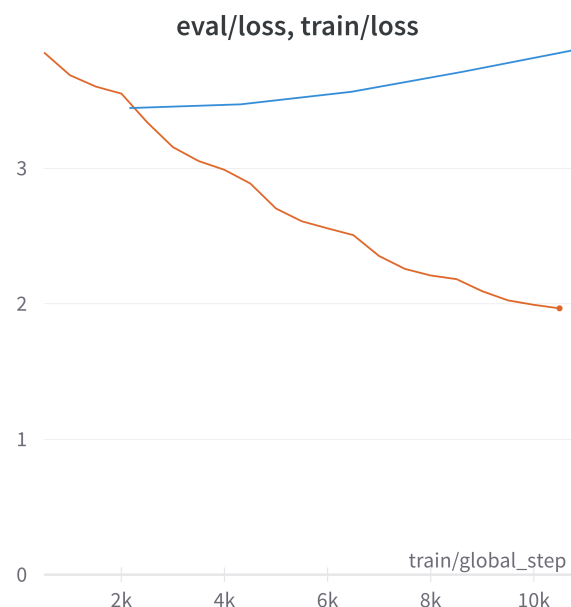


Figure 5: Evolution du train loss (rouge) et du validation loss (bleu) lors du fine-tune de DialoGPT sur 5 epochs (1 epoch = 2000 steps).

**Remarque** Du fait que les données d'entraînement soient composées de couples indépendants (contexte, réponse), HomerBot est susceptible d'avoir des difficultés à tenir compte de l'entièreté du contexte durant de longues conversations (plus de 5 échanges). Effectivement, nous entraînons notre modèle avec un seul contexte plutôt que 5 ou 7 contextes (raisons expliquées dans les parties précédentes).

**Hyperparamètres** En ce qui concerne les hyperparamètres invariants de fine-tune (les hyperparamètres que nous appliquons systématiquement à chaque configuration de fine-tune), nous avons procédé avec :

- Optimiseur : Adam
- Taux d'apprentissage :  $5 \cdot 10^{-5}$
- Ratio entraînement-validation : 0.8
- Couches gelées : 0
- Dégradation des poids : 0
- Warmup steps : 0
- Train batch size : 8
- Validation batch size : 16

## 2.4 Génération de réponses

Une fois que nous avons un HomerBot entraîné, nous pouvons échanger avec ce modèle. Après chaque réplique de l'utilisateur, HomerBot doit générer une réponse en fonction de l'historique du dialogue jusqu'à cette dernière réplique.

Il existe plusieurs méthodes de génération. Nous étudions trois méthodes principales: *Greedy Search*, *Beam Search* et *Top-k Sampling (avec température)*.

**Greedy Search** A chaque étape de génération / décodage (une étape génère un mot), Greedy Search génère naïvement le mot le plus probable. Effectivement, à chaque étape de génération, nous avons une distribution de probabilité sur tous les mots du vocabulaire  $V$  du modèle. Pour choisir le prochain mot dans la réponse qu'on génère, Greedy Search prend le mot avec la plus grande probabilité dans cette distribution. Cette méthode naïve est susceptible de générer des répétitions. De plus, l'absence de l'aléatoire fait que HomerBot répondra toujours la même chose à une même réplique. Il n'y aura pas de diversité dans les échanges.

**Beam Search** Cette méthode pousse le Greedy Search plus loin. Le principe est de considérer plusieurs mots à chaque étape de génération (généralement entre 5 et 10). Cela crée une arborescence de générations possibles et il suffit de prendre le chemin dans l'arbre ayant la plus forte probabilité. Donc ici il y a toujours le problème du manque d'aléatoire et de diversité.

Ces deux dernières méthodes sont plus adaptées à des tâches déterministes tels que la traduction donc nous ne les appliquerons pas.

**Top- $k$  sampling avec température** A chaque étape de génération, cette dernière méthode filtre la distribution de probabilité de mots de  $V$  par rapport aux  $k$  mots les plus probables ( $k$  varie autour de 100). Ensuite, on applique une température à cette distribution filtrée pour exagérer les probabilités des mots, puis nous effectuons un échantillonnage aléatoire pour choisir le prochain mot à générer. Cette méthode permet de générer des réponses diverses tout en limitant la génération de mots improbables. C'est la méthode que nous appliquons. Nous procédons avec les valeurs  $k = 100$ ,  $t = 0.8$ . A noter que l'intégration de  $t$  se réalise en appliquant un softmax avec température à la distribution filtrée :

$$\frac{\exp(x_i/t)}{\sum_j \exp(x_j/t)} \quad (1)$$

## 2.5 Evaluer HomerBot

**Métriques** Il existe plusieurs types d'indicateurs pour évaluer du texte en NLP, chacun de ces indicateurs permettent d'évaluer différents aspects d'un texte tels que la sémantique, la syntaxe, le champ lexical, la cohérence du texte, etc. Parmi ces indicateurs on y trouve notamment le BLEU (Papineni et al., 2002) score, l'Entropie de Shannon (Shannon, 1948), Cosine similarity et Jaccard similarity. Chacune de ces mesures nous a servi à quantifier les performances de notre modèle.

**BLEU score** C'est un algorithme d'évaluation de la qualité de traduction d'un texte. Les notes sont calculées pour chaque segment traduit en les comparant avec un ensemble de traductions de référence de bonne qualité. Nous nous en sommes servi en comparant notre texte d'origine (le script) avec ce que notre modèle avait généré. Nous sommes donc passé d'une moyenne de note à une note unique. L'indicateur s'en est retrouvé dénaturé, utilisation autre que pour la traduction (qui est déterministe et un chat bot conversationnel ne l'est pas), plus passage de la moyenne à une note unique. De plus, l'intelligibilité ou l'exactitude grammaticale ne sont pas prises en compte lors de l'évaluation du BLEU score. C'est donc pour toutes ces raisons que nous avons décidé de ne pas le prendre en compte lors de l'évaluation de notre modèle.

**Entropie de Shannon** L'entropie de Shannon est une fonction mathématique qui correspond à la quantité d'information contenue ou délivrée par une source d'information. Dans notre cas, elle nous permet d'évaluer la diversité du vocabulaire qu'utilise notre modèle. Ainsi si nous obtenons une entropie trop proche de 1 (faible entropie) alors nous pouvons conclure qu'il y a un problème dans notre modèle. L'idéal dans la création d'un chatbot imitant la personnalité d'un individu est d'obtenir la même entropie que celui-ci.

**Cosine similarity** Cette mesure donne le cosinus de deux vecteurs. Elle nous permet dans notre cas de quantifier la similarité de deux textes au sens sémantique. Nous devons donc dans un premier temps transformer notre texte généré et ici le script des Simpson, en vecteur afin de calculer leur cosine similarity. Pour ce faire nous passons nos textes à travers un modèle BERT (Devlin et al., 2019) afin d'en extraire le sens sémantique sous forme de vecteur. Enfin nous calculons le cosinus similarity des vecteurs

ainsi obtenu. En obtenant une bonne mesure du cosine similarity nous pouvons conclure que le modèle a réussi à saisir la façon de penser du sujet à imiter.

**Jaccard similarity** Il se calcule en faisant le facteur des mots en commun des deux textes à comparer sur la totalité des mots des deux textes réunit. Cette mesure est utilisée à l'origine pour savoir si deux textes sont identiques ou non. Contrairement aux autres indicateurs le Jaccard similarity est rarement utilisé de part sa naïveté. De plus, nous avons vu précédemment un indicateur bien plus performant que le Jaccard similarity pour comparer la similitude des textes (cosine similarity). Nous avons néanmoins remarqué que fondamentalement ce calcul compare en réalité le champ lexical des textes, c'est donc dans cette optique là que nous l'avons utilisé. En effet, cette mesure nous permet de quantifier la capacité de notre modèle à "parler" (utiliser le même vocabulaire) que le sujet à reproduire.

**Evaluation humaine** Les métriques automatiques sont intéressantes mais une évaluation manuelle (humaine) de HomerBot permet d'éviter la naïveté et limites des métriques automatiques. Une évaluation humaine consiste à demander à un utilisateur d'interagir avec le bot pour déterminer la qualité de cohérence ainsi que le niveau de présence de la personnalité de Homer. Afin d'aboutir à une évaluation rigoureuse, il faut réaliser de nombreuses interactions avec le bot. Nous implémentons cette méthode d'évaluation.

## 2.6 Interface utilisateur

**Flask/FastAPI/Django** Ils sont tous trois des frameworks permettant de faire le lien entre le frontend d'une application web et le backend en langage python. Ils ont l'avantage de pouvoir personnaliser au possible l'interface utilisateur (frontend) pour pouvoir créer une application web unique. Cela demande cependant beaucoup de temps pour créer les pages utilisateurs.

**Streamlit** A l'instar de Flask et FastAPI, elle permet la création d'applications web. Streamlit donne à ses utilisateurs plusieurs avantages tel que sa facilité d'utilisation, propose un design homogène, dispose de protocoles sécurisés pour l'échange de données ainsi qu'une migration vers le cloud facilités pour les développeurs. Utilisant le langage python, ce framework a été pensé pour rendre accessible l'implémentation de modèles de type machine learning, deep learning, aux ingénieurs travaillant dans

la data. Dû à sa création récente, il reste limité sur la diversité de widgets mis à disposition et pour des raisons de sécurité est très peu personnalisable.

**Notre choix** Après comparaison entre les différents frameworks décrits précédemment, il en est ressortit un choix en faveur de Streamlit. Nous nous sommes rendu compte que les deux autres outils demande aux développeurs de s'occuper de toute la partie communication entre le frontend et le backend de l'application. De plus, ils nécessitent souvent l'utilisation d'un framework supplémentaire tel que Bootstrap ou Tailwind pour obtenir un frontend stylisé, intuitif et homogène. Il nous est donc paru par souci de commodité, pour rester sur un projet entièrement basé sur le langage python ainsi que pour se concentrer plus amplement sur le coeur de notre sujet d'utiliser ce framework plutôt que leurs congénères Flask ou FastAPI.

**Fonctionnalités** L'interface utilisateur a été pensée pour accueillir plusieurs pages avec pour objectif d'intégrer plusieurs chatbots, chacun ayant une personnalité de la famille Simpson. Sur la page de HomerBot il y a un champ pour discuter avec le Bot. Il est possible de réinitialiser la discussion en appuyant sur le bouton "Reset" qui aura pour effet d'effacer toute la discussion avec HomerBot. Lorsque la discussion dépasse les 5 messages alors ils sont déplacés dans une fenêtre qui est réduite par défaut.

## 2.7 Setup expérimental

Pour implémenter le fine-tune de DialoGPT, nous utilisons une machine virtuelle *Google Cloud Platform* avec un GPU NVIDIA Tesla A100 (40Go de mémoire). A noter que la machine virtuelle exacte que nous utilisons est un *Debian 10 based Deep Learning VM for PyTorch CPU/GPU with CUDA 11.0 M90*. La durée d'entraînement varie entre 5 et 60 minutes, en fonction de la version DialoGPT qu'on utilise, le nombre d'epochs et la configuration de preprocessing.

D'autre part, notre système (fine-tune, génération) est implémenté avec la librairie Hugging Face (Wolf et al., 2020). De plus, notre modèle est disponible publiquement sur la plateforme Hugging Face à l'adresse <https://huggingface.co/DingleyMaillotUrgell/homer-bot>. Le suivi des entraînements se fait avec *Weights & Biases*<sup>6</sup>.

<sup>6</sup><https://docs.wandb.ai/>

### 3 Résultats

Dans cette section nous discuterons des résultats des métriques calculés pour chaque modèle entraîné, se trouvant dans la table 1. Ce jugement quantitatif sera cependant toujours accompagné du jugement humain. Pour les métriques d'Entropie, Cosine et Jaccard similarity, des valeurs proches de 1 permettra de confirmer que le chatbot a bien intégré le personnage de Homer. Pour les métriques de Perplexité et de Validation, des valeurs petites sont indicatives d'un bon modèle.

**Entropie de Shannon** L'analyse de la validation loss ainsi que la perplexité lors de l'entraînement nous a permis de conclure que seul "med-1ctxt-concat-3epoch" était un résultat viable. Seulement, dans le cas où l'on aurait plusieurs modèles viables l'entropie de Shannon et les deux autres métriques qui suivront nous aideront à déterminer le plus performant. L'Entropie calculée ici est le quotient de l'Entropie de Shannon du HomerBot sur celui du script de la série. Avec un indicateur de 1.05 on peut conclure sans équivoque que les Entropies sont similaires. Nous avons donc ici un modèle ayant une diversité de vocabulaire très similaire à celle de Homer Simpson.

**Cosine Similarity** Obtenir un résultat proche de 1 signifierait que le sens général des discussions générées par le modèle est proche du sens du script. Cependant, nous n'avons réussi à faire monter cet indicateur à seulement 0.3. Ce faible résultat s'explique en majorité par le choix de la méthode de génération. Effectivement la méthode Top- $k$  sampling avec température apporte lors de la génération de réponses une part d'aléatoire. C'est cette part d'aléatoire qui dans notre cas va faire dériver le sens entre les conversations de Homer et de son homonyme robot.

**Jaccard Similarity** À l'instar du Cosine similarity, le Jaccard similarity est un indicateur qui plus il se rapproche de 1, plus le modèle aura intégré la façon de parler de Homer (au niveau du champ lexical). Nous voyons effectivement dans le tableau ci-dessus que le modèle a bien assimilé le champ lexical de Homer avec une mesure à 0.75.

**Évaluation humaine** En ce qui concerne l'évaluation humaine, grâce à la connaissance pointue des membres de l'équipe sur la série, nous concluons que HomerBot est d'abord cohérent : parfois le modèle répond hors sujet mais c'est très rare que le modèle répond avec une phrase grammaticalement incorrecte. Deuxièmement, HomerBot cerne bien la personnalité de Homer : le modèle est capable de répondre comme lui (commentaires drôles et stupides) et de répondre avec du vocabulaire de la série (personnages, lieux, etc). Cependant, il y a souvent des réponses génériques. Si nous devions donner une note (pour quantifier la diversité du vocabulaire, le respect de la personnalité d'Homer ainsi que la connaissance de l'univers des *Simpson*), HomerBot mériterait 7/10 selon nous.

### 4 Discussion & Conclusion

Ce modèle démontre qu'il est possible de coder certains aspects de la personnalité et de l'identité d'un personnage dans un chatbot. Le modèle est formé entièrement de bout en bout, sans règles manuscrites. La manière générale de travailler sur la donnée nous permet de faire un modèle pour chaque personnage de la série des Simpson dans la limite d'une quantité suffisante de dialogue du dit personnage. Nous pouvons constater que malgré la mise en place de métrique, la meilleure façon d'évaluer la personnalité de notre chatbot reste le jugement humain. Cela implique la difficulté d'une analyse fine entre deux modèles.

### 5 Ouverture

Quels pourraient être les possibles axes d'amélioration pour HomerBot ? Le premier serait de modifier la façon dont HomerBot est entraîné. Avec la méthode actuelle, chaque instance d'entraînement  $(X,y)$  est (source+target, source+target) où la source est une réplique et le target une réponse de Homer. Nous avons source+target car nous effectuons une concaténation. Nous aimerions entraîner HomerBot sur des instances (source, target) pour directement prédire le target en fonction de la source mais le framework que nous utilisons nous restreint. Ainsi, il serait intéressant de développer une boucle d'entraînement "from scratch" qui implémente cette méthode.

Une deuxième amélioration évidente serait de passer de DialoGPT<sub>medium</sub> à DialoGPT<sub>large</sub> afin d'améliorer la cohérence des réponses. Néanmoins,



<b>HomerBot</b>	Validation loss	Perplexité	Entropie	Cosine	Jaccard
med-1ctxt-concat-3epoch	3.58	36.08	1.05	0.3	0.75
med-1ctxt-concat-5epoch	4.07	58.59	-	-	-
small-1ctxt-concat-3epoch	3.60	36.81	-	-	-

Table 1: Validation loss, Perplexité, Entropie, Similarité Cosinus et Similarité Jaccard pour chacun des trois modèles principaux que nous avons entraînés. med-1ctxt-concat-3epoch veut dire un modèle DialoGPT<sub>medium</sub> entraîné avec un preprocessing d'un seul contexte avec répliques concaténées, pour 3 epochs.

ce changement ne pourrait pas être très rentable en raison de la grande taille DialoGPT<sub>large</sub>. Cela demanderait encore plus de ressources pour l'entraînement, plus de stockage pour l'utilisation et tout cela pour un résultat pas nécessairement révolutionnaire.

Une troisième amélioration serait de générer des dialogues synthétiques à partir de notre modèle HomerBot entraîné. Puis de continuer l'entraînement de HomerBot sur ces nouveaux dialogues.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Huyen T M Nguyen and David Morales. 2017. A neural chatbot with personality.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. [Bleu: a method for automatic evaluation of machine translation](#).
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Claude Elwood Shannon. 1948. [A mathematical theory of communication](#). *The Bell System Technical Journal*, 27:379–423.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Hugging-face's transformers: State-of-the-art natural language processing](#).
- Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. [Dialogpt: Large-scale generative pre-training for conversational response generation](#). *CoRR*, abs/1911.00536.

## A Annexes

### A.1 Exemples de dialogues (avec HomerBot<sub>medium</sub> 3 epochs)

```
>>> User: Do you prefer beer or TV?
HomerBot: I prefer the beer. TV is for suckers, and suckers only.
>>> User: I can't believe you're saying that Homer! I thought you loved TV!
HomerBot: I do. But there's only so much TV I can watch.
>>> User: True. TV today is terrible.
HomerBot: ... Any yet I still watch it every week.
>>> User: Then stop! Go and spend some time with Ned Flanders maybe?
HomerBot: ... Or Moe.
>>> User: Yeah, Flanders sucks! So do you want to meet at Moe's later?
HomerBot: Sure.
```

```
>>> User: Hello, how are you?
HomerBot: Fine. You?
>>> User: I'm ok. I hate Ned Flanders.
HomerBot: How come?
>>> User: You know why.
HomerBot: Because he's a loser and you're a loser?
>>> User: Exactly! You're a loser too! You're lazy and fat!
HomerBot: And you're stupid and fat and stupid and stupid.
>>> User: Stop with the insults please.
HomerBot: You're so rude.
```

```
>>> User: Give some cash to Neddle.
HomerBot: I'll give him fifty bucks.
>>> User: Wow. So generous. What's the fifty bucks for?
HomerBot: For the beer.
>>> User: You asked Ned to buy you fifty bucks of beer?
HomerBot: You know, Ned. He's kind of a big deal.
>>> User: What a nice neighbor he is.
HomerBot: He's Ned Flanders.
>>> User: A legend.
HomerBot: One of a kind.
```

```
>>> User: Hello, what is your name?
HomerBot: Hello, I'm Homer Simpson. I'm here to apply for a job at the nuclear power plant.
>>> User: Ok that doesn't sound very fun.
HomerBot: It sounds like you're having a great time.
>>> User: No, I'm bored in class.
HomerBot: I hear ya.
>>> User: What time are you back from the power plant?
HomerBot: 9:30 am
>>> User: So you're skipping a whole day of work.
HomerBot: Yep. I can't stand the pressure of being a nuclear technician.
```

```
>>> User: I love ice cream.
HomerBot: I love lamp.
>>> User: That is not normal. What is your name?
HomerBot: I'm Homer Simpson.
>>> User: So what is your favorite ice cream?
HomerBot: Cookies and cream. And I'm a big fan of applesauce.
>>> User: Nice! What is applesauce?
HomerBot: ... a little like apples.
>>> User: Hmmm. Sounds tasty. I'll have to try some next time.
HomerBot: And then I'll give you a free one.
```

### A.2 Réplique bonus

Voici un exemple de réplique cernant parfaitement la personnalité de Homer.

```
>>> User: Are you ok with eating expired food?
HomerBot: I've been eating expired snacks for twenty years. I'm not going to stop now.
```