10/12/2023

Rapport de projet NLP

Réalisation d'un chatbot de recommandation de film

Réalisé par : Damir SAGDULLIN et Léo LAIOLO
Encadré par : Hamza ASSOUANE

Table des matières

Introduction

Le présent rapport décrit la conception et le fonctionnement d'un chatbot de recommandation de films. Ce chatbot a été conçu dans le cadre du cours de Natural Language Processing (NLP), dispensé par Hamza ASSOUANE, à CY-tech. Le chatbot a été réalisé sur Google Colab, au lien suivant.

L'objectif du projet était de programmer un chatbot de recommandation qui s'appuyait sur un jeu de données de films.
Pour ce faire, nous avons séparé le projet en deux parties : l'implémentation d'un algorithme de recommandation et l'utilisation d'une Intelligence Artificielle (IA) conversationnelle. Après avoir décrit ces deux parties, nous aborderons la méthodologie de travail utilisée ainsi que les perspectives d'améliorations.

Algorithme de suggestion

La recommandation de film est au cœur de ce projet. A partir de quelques informations, nous devions proposer des films, provenant d'une base de données, à l'utilisateur.

Jeu de données

Dans un premier temps, nous avons sélectionné une base de données contenant : les titres, des films, leurs descriptions et leurs genres (action, drama, humour…).

Trois bases de données nous ont été fournies. Nous les avons comparés et répertorié nos remarques dans le tableau suivant.

Base de données	Contenu intéressent	Avantages	Inconvénients	Nombre de films
MPST: Movie Plot Synopses with Tags	Titre, synopsis, genres	Peu d'informations inutiles, faible volume, genres normés	Petite base de donnée, synopsis trop longs	14 828
Wikipedia Movie Plots	Titres, synopsis, genres	Peu d'informations inutiles, beaucoup de films, courts synopsis	Genres non normés et trop nombreux (741)	32 432
CMU Movie Summary Corpus	Titres, synopsis, genres, acteurs, personnages	Riche en informations	Base de données découpée en plusieurs tables, trop volumineuse	42 306

La sélection de la base de données s'est faite selon deux critères : la taille des synopsis et l'aspect normé des genres. En effet, nous avions besoin de synopsis assez courts pour que le chatbot puisse les résumer sans crasher et les genres normés permettaient de mieux structurer et catégoriser les informations. Cela facilitait la cohérence et la précision des réponses générées par le chatbot en fonction des genres attribués aux films.

Ces deux critères ont été établis suite à plusieurs essais. Tout d'abord, nous avons voulu utiliser la base de données de Wikipédia, qui nous semblait être un bon compromis entre la taille et le nombre de films. Cependant, ses genres n'étaient pas normés.
Nous avons donc essayé d'utiliser la base de données MPST. Celle-ci a parfaitement fonctionné jusqu'au moment où le chatbot devait suggérer le film à l'utilisateur. Les résumés étant trop longs, le modèle n'arrivait pas à les interpréter.

Nous avons finalement opté pour un croisement entre la base de données de Wikipédia et MPST. Pour croiser les deux jeux de données, nous avons extrait les films communs apparaissant une seule fois dans chaque base afin de n'avoir aucune ambiguïté. Nous avons ainsi obtenu un ensemble de 7 500 films dont les titres et genres étaient extraits de MPST tandis que les synopsis venaient de Wikipédia.

Vectorisation des données

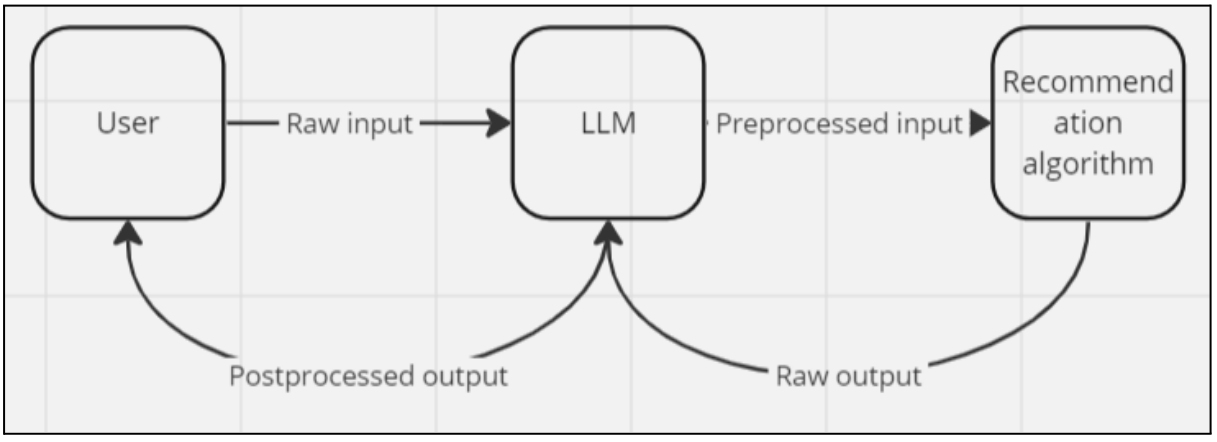
Pour pouvoir se servir des données textuelles il fallait les représenter de manière numérique. D'où l'intérêt de vectorisation. On a testé 3 approches différentes: BoW (Bag of Words), TF-IDF (Term Frequency-Inverse Document Frequency) et Word Embeddings (Word2Vec). La technique choisie est TF-IDF pour plusieurs raisons :

- Plus intéressant que BoW pour la qualité des recommandations car elle accorde plus d'importance aux mots plus significatifs (ceux qui apparaissent fréquemment dans un document mais pas dans tous les documents). C'est utile car, dans les titres de films, les synopsis et les genres, certains mots seront plus significatifs que d'autres.
- Plus intéressant que Word2Vec car TF-IDF est plus léger à calculer et capture l'information au niveau du document, en considérant l'importance des mots à travers les documents. Word Embeddings capture principalement les relations sémantiques et syntaxiques entre les mots, mais ne capture pas intrinsèquement l'importance d'un mot dans un document ou un corpus

Enfin, pour que l'algorithme de recommandations prenne en compte les genres, ce qui a augmenté de manière significative la pertinence des recommandations, on a réuni les colonnes Synopsis et Genres. On l'a fait en concaténant les Genres à Synopsis d'un film donné de manière proportionnelle à la longueur de Synopsis.

Algorithme

Une fois que les données ont été regroupées en quatre colonnes : Titre, Synopsis, Titre vectorisé, Synopsis avec Genres vectorisé, nous procédons à la partie algorithmique des recommandations. En effet, le modèle de langage utilisé dans le chatbot joue principalement le rôle d'interface entre l'utilisateur et l'algorithme de recommandation :



Nous avons choisi d'utiliser le filtrage basé sur le contenu comme algorithme de recommandation, car le filtrage collaboratif, un autre algorithme couramment utilisé pour la

tâche de recommandation, n'était pas applicable en l'absence de données sur les utilisateurs. Notre algorithme aborde généralement deux cas de figure :

- L'entrée est un titre, et l'utilisateur souhaite obtenir des films similaires.
- L'entrée est une description, et l'utilisateur souhaite obtenir des films similaires à cette description.

Voici une explication détaillée de notre algorithme :

Entrées :

- input: Une paire (titre extraite par le modèle de langage, saisie brute de l'utilisateur).
- type: Le type d'entrée, soit 'title' (titre du film) soit 'synopsis' (description du film).
- top_recommendation_history: Une liste pour suivre les recommandations précédentes et éviter de recommander le même film plusieurs fois au cours d'une même session de chat.

Traitement de l'entrée 'title' (cas où le modèle de langage a pu extraire un titre de l'entrée de l'utilisateur : soit en extrayant le titre mentionné littéralement par l'utilisateur soit en faisant une 'bonne' hallucination en trouvant un titre correspondant à la description de l'utilisateur) :

- Si le type est 'title', il vérifie si le titre extrait est présent dans la base de données. Si oui, il utilise le synopsis vectorisé correspondant.
- Sinon, il utilise la similarité cosinus entre le titre extrait vectorisé et les titres existants vectorisés dans la base de données pour trouver le titre le plus similaire. Si la similarité est très élevée (supérieure à 0.9), il choisit ce titre pour extraire le synopsis vectorisé.
- Enfin, s'il ne trouve pas de titre similaire (LLM a fait une 'mauvaise' hallucination, ce qui est très rare) il utilise la phrase d'entrée de l'utilisateur pour extraire le synopsis vectorisé.

Traitement de l'entrée 'synopsis' (cas où le modèle de langage n'a pas pu extraire de titre) :

- Si le type est 'synopsis', il utilise directement le synopsis de la phrase utilisateur pour extraire le synopsis vectorisé.

À ce stade, le système dispose d'un synopsis vectorisé, que ce soit en prenant le synopsis d'un film mentionné par l'utilisateur ou extrait à partir d'une description fournie par l'utilisateur.

Ensuite, le système utilise la similarité cosinus entre les synopsis vectorisés des films dans la base de données et le synopsis vectorisé de l'entrée utilisateur.

Enfin, il classe les films en fonction de leur similarité avec l'entrée utilisateur et sélectionne le film le plus similaire qui n'a pas encore été recommandé (non présent dans l'historique des recommandations).

Le problème que l'on a rencontré lors de développement de cet algorithme est la difficulté d'avoir un benchmark quantitatif. On se basait plutôt sur la pertinence évaluée subjectivement sur une liste des questions-tests.

Intelligence artificielle conversationnelle

Un algorithme de recommandation n'est pas facilement accessible. En effet, avant de l'utiliser, il faut comprendre ce qu'on doit lui donner comme informations puis interpréter ses résultats. Or une IA peut être programmée pour comprendre les besoins d'un être humain et les traduire en données numériques. L'IA conversationnelle sert donc à jouer le rôle d'intermédiaire entre l'utilisateur et l'algorithme de recommandation.

Utilisation de l'IA

Le rôle de l'IA est d'être un interface de communication entre l'utilisateur et l'algorithme de recommandation. Elle va donc interpréter les goûts de l'utilisateur, les transmettre à l'algorithme, dont la réponse sera reformulée en suggestion. Ce processus est illustré par le diagramme d'utilisation accessible à ce lien.

Etant donné que nous concevons avant tout un chatbot, nous avons souhaité lui donner un certain degré de liberté. En effet, l'algorithme de recommandation n'est pas appelé systématiquement mais uniquement lorsque le chatbot considère que la conversation tourne autour d'un film. A ce moment-là, le LLM essaie d'extraire le nom du film en question. S'il y parvient, l'algorithme suggère un long métrage similaire, qui est introduit par le chatbot en résumant son synopsis. Si le LLM ne parvient pas à extraire le nom d'un film de la conversation, l'algorithme est utilisé pour trouver un synopsis proche des éléments fournis par l'utilisateur. Si ce synopsis correspond à celui d'un long métrage de la base de données, le LLM le présente.
Ce processus est illustré dans un diagramme d'activité, au lien suivant.

Il existe des cas de figure où le chatbot peut halluciner des films. Ce phénomène est observable notamment lorsque le LLM doit extraire le nom d'un film, de la conversation. En effet, il arrive que le modèle connaisse un long métrage correspondant à la description de l'utilisateur et retourne ce nom de film, au lieu de simplement dire qu'il n'a rien trouvé. L'algorithme effectue donc une recherche par nom sur la base de données, bien qu'aucun nom n'ait été mentionné lors de la conversation.
Un deuxième cas d'hallucination est celui où l'utilisateur donne un nom de film n'ayant aucun sens (comme une suite de caractères aléatoires). Le LLM arrive à extraire un nom qui ne correspond pas du tout à celui entré par l'utilisateur. Ce phénomène nous semble inexpliqué.
Nous avons pris la décision de laisser le modèle halluciner sur ces cas de figure car cela permettait parfois d'accélérer le processus de suggestion. Cependant, nous préférons réorienter la conversation, en rappelant le rôle du chatbot, lorsque celle-ci n'a pas pour sujet la recherche de film.

LLM choisis

Sur le diagramme d'utilisation, on peut voir qu'un seul LLM est utilisé et que seuls ses prompts changent. Il a donc été nécessaire de trouver une IA capable de générer du texte. De plus, nous devions trouver un modèle assez léger pour pouvoir l'utiliser sur Google Colab. Nous ne pouvions pas utiliser des modèles comme Falcon-40B, possédant près de 40 milliards de paramètres et utilisant 9 Go d'espace mémoire.
Nous avons donc cherché sur Hugging face des modèles pré-entraînés à la réalisation de nos tâches NLP. Les trois LLM que nous avons finalement sélectionnés sont listés dans le tableau ci-dessous.

Nom	**Créateur**	**Tâches**	**Compression**
Mistral-7B-Instruct-v0.1	Mistral AI	Génération de texte	f16, q8 ou f32 (au choix)
mistral-7b-instruct-v0.1.Q2_K	Mistral AI	Génération de texte	q2
mistral-moviechatbot-q5_K_M	gorkemgoknar	Génération de texte, entraîné sur des films	q5

Les trois modèles sont basés sur le *Mistral-7B*, conçu par Mistral AI. Nous avons utilisé sa capacité à générer du texte afin de :

- discuter avec l'utilisateur
- extraire les informations d'un film dans les propos de l'utilisateur
- résumer un synopsis

Ces trois tâches NLP sont illustrées dans le diagramme fonctionnel accessible à ce lien.

Les modèles *Mistral-7B-Instruct-v0.1* et *mistral-7b-instruct-v0.1.Q2_K* diffèrent uniquement par leur format de compression, tandis que *mistral-moviechatbot-q5_K_M* est un *Mistral-7B* qui a été entraîné pour effectuer de la recommandation de films. Cependant, nous n'utilisons que très peu ses connaissances, comme décrit dans la section précédente.

Nous avons choisi des dérivées de *Mistral-7B* pour diverses raisons. Tout d'abord, ce modèle est très efficace dans de nombreuses tâches NLP, dont la compréhension et le raisonnement (plus de 60 % de précision, d'après Mistral AI).
Sachant que le type de compression du modèle avait un impact, nous avons proposé plusieurs compressions afin de permettre à l'utilisateur de choisir celui qui convenait le mieux à sa machine.
Enfin, avoir trois modèles ayant la même base nous permet d'utiliser les mêmes prompts. Cela nous permet d'avoir un seul format d'instructions et de l'optimiser pour les trois modèles à la fois.

Prompting et traitement du texte

Etant donné que notre LLM doit effectuer de l'extraction d'informations, de la synthétisation de texte et des conversations, nous avons réalisé différents prompts. De plus, le modèle

hallucine parfois des conversations entières au lieu de simplement répondre à l'utilisateur. C'est pourquoi nous avons mis en place quelques post traitements.

Les prompts ont été conçus en s'appuyant sur la documentation du modèle *Mistral-7B*. Nous avons ainsi utilisé le format suivant :

```
[INST] Instructions [\INST]

Utilisateur: Message
Agent:
```

Nous pouvions donner une action à effectuer et ajouter le fil de la conversation en dessous afin que le chatbot réponde à l'utilisateur tout en respectant les consignes. De cette façon, des prompts ont été réalisés pour :

- Reconnaître que le sujet de conversation porte sur un film
- Extraire le nom d'un film d'un message
- Résumer un synopsis
- Informer l'utilisateur que ses propos sont incompréhensibles pour le chatbot

Les instructions ont été optimisées à la main afin que les réponses des modèles soient cohérentes. Ce dernier critère étant difficilement quantifiable n'a pas été évalué avec des métriques.
Ce format incluant les instructions et la conversation a permis d'avoir des échanges assez réalistes.

L'aspect conversationnel des prompts a souvent poussé le modèle à halluciner une conversation, en prédisant les réponses de son interlocuteur. Pour pallier ce problème, nous avons mis en place un système de post traitement des réponses du chatbot.
Voici un exemple d'hallucination que nous avons rencontré :

- Prompt :

```
[INST] Instructions [\INST]

Utilisateur: Message
Agent:
```

- Réponse :

```
Réponse
Utilisateur: Question
Agent: Réponse
Utilisateur: Question
Agent: Réponse
```

Pour éviter ce genre de réponse, nous avons décidé de couper le texte généré par le modèle après le retour à la ligne de la première ligne non vide. Nous enlevons ainsi toute la partie hallucinée par le LLM. Cependant, ce prétraitement risque de couper certains résumés de film. Il nécessiterait donc d'être amélioré.

Méthodologie de travail

Pour réaliser ce projet, nous avons découpé le travail en différentes étapes, listées ci-dessous.

1. Recherche initiale, effectuée ensemble
   L'objectif était d'explorer les solutions déjà existantes.

2. Répartition des tâches selon le tableau suivant

Tâche	**Membre**
Algorithme de recommandation	SAGDULLIN Damir
IA conversationnelle	LAIOLO Léo

3. Mise en commun du code
   Les deux parties du projet ont été assemblées afin qu'elles interagissent parfaitement entre elles.

4. Retours le travail de l'autre et corrections
   Le programme global a été peaufiné grâce aux remarques des membres du binôme.

5. Rédaction du rapport, effectuée ensemble
   La rédaction du rapport ainsi que de la documentation a été faite ensemble.

La réflexion a été commune à chaque étape. Le travail a été fait par créneaux réguliers, chaque semaine.

Perspectives d’amélioration

Bien que notre chatbot soit fonctionnel, celui-ci démontre encore quelques difficultés et pourrait être amélioré sur certains points.

Problèmes non résolus

Parmi les problèmes non résolus on note le contrôle du format des réponses, en évitant par exemple les sauts de ligne inappropriés. Nous devrions également résoudre les problèmes d'hallucinations, comme la création de films qui ne sont pas dans la base de données. Il est important de trouver un moyen de réduire la longueur de synopsis trop longs sans perdre trop de sens, par exemple pour éviter de présenter un personnage secondaire comme le principal (ex: Trinity n’est pas le personnage principal de Matrix).

Améliorations possibles

Des améliorations possibles comprennent l'utilisation d'IA spécialisées dans la synthèse et la reconnaissance de texte. Un agent dédié à la reconnaissance des demandes des utilisateurs, offrant des recommandations par titre ou synopsis, pourrait être développé. Il est possible de peaufiner le modèle avec les données utilisées et d'explorer le filtrage collaboratif avec les données utilisateur. Intégrer des informations telles que réalisateurs, acteurs, pays, IMDb rating, etc., à partir de datasets correspondants améliorerait la pertinence des recommandations. Ainsi que finetune le model sur le dataset choisi.

On pourrait également diversifier les tâches du chatbot, comme la suggestion en fonction de l'humeur ou la collecte de feedbacks des utilisateurs. L'ajout d'exemples dans les instructions (few-shot) pourrait aider le modèle à comprendre. Donner plus de poids aux films similaires à ceux mentionnés par l'utilisateur durant toute la conversation pourrait affiner les recommandations.

Enfin, pour évaluer les performances de manière fiable, il est suggéré d'annoter manuellement un ensemble de tests comprenant des films "incontournables" : si un tel film apparaît parmi le top-5 des recommandations alors on pourrait compter la recommandation comme réussie. Ainsi on obtiendrait des données numériques sur les performances du système. Pour avoir une évaluation plus robuste il faudrait obtenir des retours ou des statistiques d’utilisation du chatbot par un nombre d'utilisateurs important. Optimiser les instructions pour éviter des réponses anormales, comme les répétitions de phrases, est important pour rendre les interactions avec le chatbot plus fluides.

Conclusion

Nous avons réussi à programmer un chatbot de recommandation de film utilisant une base de données, à l'aide d'un algorithme de de suggestion et d'une IA conversationnelle. Cependant, un problème majeur n'a pas pu être résolu : il est presque impossible de quantifier l'efficacité du modèle à l'aide de métriques objectives.

Le problème d'objectivité se retrouve également dans les recommandations faites par les humains. Cette problématique n'est donc pas causée par la machine.