Projet TAL - Compte-rendu

Adrien LAVILLONNIERE - Corentin MANSCOUR - Hien Minh NGUYEN

5 mai 2018

Table des matières

Chapitre 1

Introduction

1.1 Contexte du projet

Ce projet a pour but de développer un chatbot racontant une histoire. Il doit être capable de comprendre les entrées des utilisateurs qui peuvent poser des questions à propos du scénario, des personnages, de l'environnement, des objets utilisés...

En utilisant plusieurs modes, qui correspondent à plusieurs "niveaux d'intelligence", le chatbot est en mesure de répondre à la plupart des questions que l'on peut lui poser. Il possède une base de données de réponses prédéfinies associées à des tags.

Quant à l'histoire, elle est prise en charge par des algorithmes fonctionnant en arrière-plan, où les actions des personnages sont gérés par la machine. L'utilisateur peut également influencer ces choix s'il le souhaite, et changer le cours de l'histoire.

1.2 Objectif du projet

Notre objectif est de réaliser un chatbot entièrement fonctionnel avec une interface basique, qui permettrait à un utilisateur quelconque de comprendre et d'interagir avec l'histoire.

Le chatbot utilise une reconnaissance par mot-clés : grâce à une liste de mots-clés prédéfinis, associés à des tags, il est possible de reconnaître le sujet d'une phrase. Le chatbot choisit ensuite la réponse la plus appropriée aux tags de la phrase en question. Les réponses doivent être suffisamment différentes pour ne pas ennuyer l'utilisateur, c'est pourquoi nous avons prévu d'en écrire au moins une centaine.

1.3 Choix du sujet

Nous avons voulu aller au-delà du concept de chatbot conversationnel basique et simuler un système plus complexe. Ce chatbot pourra donc mener à des situations intéressantes grâce à l'ajout d'une dimension dynamique (le système prenant en charge l'avancée de l'histoire).

Nous avons choisi de coder un chatbot parlant anglais par soucis de simplicité : en effet, nous pensons que le français est plus complexe en terme de conjugaison et de syntaxe, difficulté que nous n'aurons pas à gérer en anglais.

Chapitre 2

La conception du chatbot

2.1 Méthodologie

Lorsque nous avons commencé notre travail, nous nous sommes basés sur les TP réalisés précédemment dans ce cours, concernant le pré-traitement de texte et la reconnaissance des spams. Nous avons choisi d'utiliser la plateforme NLTK qui offre en particulier la fonction tokenize : son utilisation sera explicitée plus loin dans ce rapport.

2.1.1 Distribution des tâches

L'utilisation de GitHub nous a également très vite permis de paralléliser le travail. Voici comment nous nous sommes répartis les tâches :

- Adrien (*Veados*) : réalisation des modes 1 et 2 du chatbot et de la base de donnée de tags
- Corentin (*neofoetus*) : réalisation du mode 3, conception du moteur simulant l'avancée de l'histoire et écriture des réponses du chatbot
- Minh (*shipanda01*) : réalisation du mode 2 et surtout 3, conception de la reconnaissance des expressions régulières

Tous les membres de l'équipe ont contribué par ailleurs à l'écriture du rapport. Le code circulant quelquefois à travers d'autres voies que github, les push réalisés par chaque membre ne sont pas forcément représentatif de leur travail.

2.2 Le chatbot

2.2.1 Fonctionnement du chatbot

Le chatbot peut se lancer dans le shell simplement grâce à la commande `python main.py`. Le téléchargement de packages NLTK peut être nécessaire avant le premier lancement. L'utilisateur est alors face à une introduction basique sur le chatbot. Il est amené à faire un choix entre :

- Lancer le programme
- Afficher des instructions suivi d'une description brève de l'histoire
- Des explications en détail sur les 3 modes

Arrivé sur l'écran de conversation, le chatbot présentera la situation et l'histoire. L'utilisateur pourra alors dire des choses au chatbot, comme : "Hello chatbot !" "What can you tell me about the story ?" "Where is Stephanie ?" etc.

2.2.2 La simulation

Le chatbot simule en arrière-plan les paramètres de l'histoire : tous les 5 input de l'utilisateur, il y a une incrémentation du temps. Le chatbot peut décider d'incrémenter le temps plus rapidement et l'utilisateur peut également en faire la demande.

Les différentes questions que l'on peut lui poser concernent l'état des personnages, leurs actions, leur inventaire... L'utilisateur peut également faire des propositions au chatbot qui pourra les accepter ou non. Plus d'informations concernant la simulation peuvent être obtenues dans le readme présent sur le répertoire git du projet.

2.2.3 Structure générale du programme

Nous avons choisi de diviser le programme en plusieurs fichiers :
— main.py : le programme principal
— parser.py : les fonctions de tokenization des inputs de l'utilisateur
— data.py : le traitement des données du dossier data et la création des dictionnaires
— compute.py : le calcul des réponses du bot
— story.py : le moteur simulant l'histoire

Les données du chatbot sont écrites en dur dans les fichiers .txt contenus dans le dossier data. Afin de pallier la mauvaise orthographe et les fautes de frappes, nous avons pris en compte les erreurs éventuelles des utilisateurs. Par exemple, l'un des fichiers contient le mot `Tommy Wiseau` mais également le mot `Tomy Wiso`.

Le parsage des entrées de l'utilisateur se fait à l'aide des fonctions du fichier tokenize.py : afin de rendre le texte interprétable par le chatbot, nous le faisons passer la fonction tokenize de NLTK. Celle-ci possède un avantage par rapport à notre propre tokenizer codé en TP : en effet, elle est capable de reconnaître les contractions et les négations (I'm, I can't...). Le chatbot fait donc notamment la différence entre I'm et I am.

2.2.4 Les différents modes

Le chatbot possède 3 "niveau d'intelligence". Lorsque nous traitons la phrase de l'utilisateur, nous privilégions d'abord le mode 3 qui est le plus intéressant et le plus complexe. Dans le cas où le mode 3 ne peut pas répondre à la question posée, c'est le mode 2 qui prend le relai et ainsi de suite.

Mode 3 C'est le mode principal où l'on peut poser des questions au chatbot et obtenir des réponses concernant l'histoire. Le principe du mode 3 repose sur une série de dictionnaires : un dictionnaire contenant les mots des champs lexicaux de l'histoire, un dictionnaire contenant les listes de tags pouvant mener à une réponse du bot, et enfin un dictionnaire contenant les réponses du bot. Ces dictionnaires sont stockés dans des fichiers textes dans le dossier /data/.

Voici comment nous avons procédé pour l'écriture des données du chatbot :

1. Première salve de questions hypothétiques pouvant être posées au chatbot
2. Depuis celle-ci, création d'un premier set de tags qui devront être reconnus
3. Association à chaque tag un "champ lexical" composé de synonymes ou d'expressions ayant un rapport avec celui-ci
4. Ecriture d'un set de réponses que le chatbot pourra opposer aux questions de l'utilisateur
5. Association de chaque réponse à une liste de tags

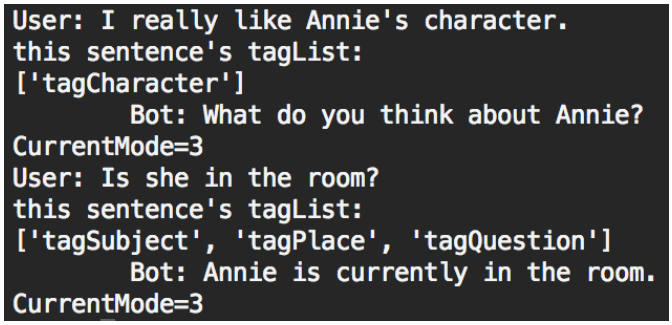

FIGURE 2.1 – Deux questions posées au bot, illustrant le fonctionnement du mode 3 et mettant en valeur le système de contexte.

6. Enfin, envoi d'une nouvelle salve complémentaire de questions au chatbot. Pour chacune de ces questions :
   - si elles sont reconnues par le chatbot, alors cela veut dire qu'il a pu la relier à un ensemble de tags déjà défini auparavant. Il n'y a alors rien à faire.
   - sinon, on enregistre la liste de tag que le chatbot nous propose pour cette question, et la relie à une de nos réponse.

Le but est d'itérer la dernière étape jusqu'à ce que le chatbot puisse répondre à une proportion satisfaisante de questions.

Les réponses sont stockées de manière suivante :
`<tag1> <tag2> {Phrase correspondant aux tags 1 et 2 avec un (joker) éventuel}`

Cela permet au chatbot d'attribuer un ensemble de tags à chaque phrase et ainsi renvoyer la bonne réponse pour une certaine série de tags.

La phrase de l'utilisateur est décortiquée par un algorithme qui en extrait les principaux tags. Une liste de tags dans n'importe quel ordre est produite. On la compare ensuite avec le dictionnaire de tags existant et on en retire une réponse si l'entrée correspondante existe. Dans le cas où aucune réponse n'est trouvée dans la base de donnée, on réduit d'un élément le nombre de tags de la liste et on effectue toutes les combinaisons possible, et on recommence le processus. Au bout de la deuxième réduction, si l'on ne trouve toujours pas de phrase correcte, le chatbot demande à l'utilisateur de reformuler. Cette limite a été implémentée afin de ne donner que des réponses qui se rapprocheraient un minimum de l'idée originale.

Voici un exemple d'une série de question que l'on pose au bot. Les sorties de déboguage ont été volontairement laissées pour illustrer le fonctionnement du bot. On remarquera que le système de contexte fonctionnement bien (le bot reconnait que l'on parle d'Annie lorsque l'on écrit 'she').

Mode 2 Comme nous avions une idée précise de ce que nous pensions faire pour le mode 3, nous avons adapté le mode 2 afin de répondre à des questions simples sur les sujets suivants : informations sur l'histoire, les personnages, les lieux, les objets et le chatbot lui-même.

Le vocabulaire de ces 5 champs sont stockés dans des fichiers texte. Les noms de ces fichiers sont utilisés pour donner un tag à chaque mot de vocabulaire correspondant au champ. Par exemple, le mot *knife* sera associé au tag *tagInventory.txt*.

Les réponses du bot sont également stockées dans des fichiers texte différents pour chaque champ lexical. Nous avons aussi programmé le fait qu'aucune réponse ne soit répétée à la suite pour rendre la conversation plus naturelle.

Mode 1 C'est le plus basique et le moins intelligent : quoique l'on puisse dire, le bot répondra par des onomatopées prédéfinies telles que "Hmm..." ou bien "Ahh..". De même que le mode 2, les réponses ne se répètent pas à la suite.

Chapitre 3

Bilan

3.1 Difficultés rencontrées et pistes d'améliorations

Des fonctionnalités manquent bien entendu dans notre chatbot. Par exemple, le mode 3 ne permet de traiter qu'une seule réponse par liste de tags. Cela a été causé par des problèmes lors du stockage des données : en effet, la façon dont les réponses du bot étaient stockées a été difficile à théoriser. Au final, nous avons choisi de coder les tags en dur pour chaque phrase de réponse, ce qui était le moyen le plus facile. Ce mode de fonctionnement est peu flexible et ne permet pas une grande variété de réponses. Nous pourrions améliorer ce système en séparant chaque réponse en morceaux de réponses qui pourront être combinées pour obtenir des phrases plus complexes et variées.

Avec plus de temps, nous pourrions implémenter entre autre :

- un système permettant de déduire le contexte d'une conversation avec plus de précision. Pour l'heure, le système en place n'est pas très perfectionné et ne permet que de retenir le nom du personnage dont il est question, d'un lieu et d'un objet. Nous pourrions faire en sorte que le chatbot parle de deux personnages en même temps,avec un système amélioré.

- la gestion de la négation multiple. Dans la version actuelle du chatbot, il existe un tag regroupant toutes les négations : celles-ci ne peuvent donc pas s'appliquer sur un nom en particulier et toute la phrase de l'utilisateur est considérée comme négative.

- un système de réponses partielles qui pourront être combinées entre elles pour former des phrases complexes. Pour l'instant, le chatbot peut répondre avec des phrases toutes faites, mais nous pourrions implémenter, grâce à NLTK, un système de réponses composées et grammaticalement correctes.

3.2 Conclusion

Ce projet nous a permis de réellement nous familiariser avec le langage Python, de découvrir ses grandes lignes et ses spécificités. Nous avons été très intéressé par le sujet et avons pris beaucoup de plaisir à explorer ses implications.

Le projet a également été très formateur en matière de forme et d'outils : l'apprentissage de git, Markdown et Latex seront sans aucun doute un grand plus dans notre carrière.

Enfin, ce projet nous a fait prendre conscience de l'importance du traitement automatique du langage naturel : utilisé dans de très nombreuses applications, c'est un domaine qui ne cessera de se développer dans le futur. Nous ne pouvons espérer qu'il évoluera jusqu'à atteindre le niveau des assistants virtuels pouvant réagir aux émotions humaines, bien connu des films de science-fiction.