ET4-Info Option TAL

Mickael Francisco SERENO
Mathieu LOUVET
Stacy GROMAT
Promo 2018

Rapport projet TAL

06 Mai 2017

1 Introduction

Le sujet qui a été choisi est celui du ChatBot (ou assistant personnel). Le rôle du ChatBot est d'avoir une "conversation" avec un ou plusieurs humains. Il existe de nombreux ChatBot, par exemple sur certains sites web, il existe des ChatBots permettant d'aider les utilisateurs. On en retrouve aussi beaucoup dans le domaine de l'intelligence artificielle, comme avec Siri de Apple. L'ensemble des membres du groupe était relativement motivé par ce sujet, celui du ChatBot.

Le thème restait toutefois à choisir : nous ne voulions pas un ChatBot trop générique, et nous sommes conscients de nos propres limites. Ainsi, nous avons voulu faire un ChatBot autour du jeu "Donjons et Dragons". Son rôle sera de se comporter comme un personnage de l'univers du jeu, qui pourra répondre à quelques questions basiques portant sur l'univers.

2 Répartition des tâches

Au tout début du projet, nous avons réfléchis à trois pour voir comment le Chatbot serait modélisé et comment il fonctionnerait. Nous pouvons dire que le rapport est le fruit de nos trois réflexions.

Ensuite nous avons eu globalement trois tâches :

— Récupérer les corpus ;

— Convertir les corpus ;

— Coder le ChatBot en partant des corpus convertis ;

Ainsi, Mathieu a récupéré les corpus via un script python, Mickael les a converti et Stacy à codé le fichier projet.py pour avoir un ChatBot. Quand le ChatBot a été codé, il fallait attendre que tous les corpus soient correctement traités. Cela demandait du temps car nous ne connaissions pas encore NLTK.

Une fois les corpus convertis, nous avons pu tester le programme et nous l'avons débogué à trois, l'avons "amélioré" petit à petit (quelques petits bugs subsistaient). Il est difficile à ce stade de dire qui a fait quoi.

3 Difficultés

La grande difficulté de ce projet résidait dans le corpus. En effet il fallait un corpus sous une forme particulière : des couples de questions-réponses. Cette forme a réduit le nombre de corpus utilisable et nous a donc compliqué la tâche. Nous avons donc choisi des scripts de films puisque les dialogues se rapprochaient de ce qu'on voulait.

De plus il fallait un corpus qui rende notre ChatBot le plus "humain" possible et surtout qui nous fournisse des réponses cohérentes.

4 Stratégie

La stratégie envisagée est assez simple. Elle consiste à récupérer de très nombreux corpus de textes, et d'identifier dans ces écrits les couples de questions-réponses. Ensuite, une question est attendue de la part de l'utilisateur. Cette question est analysée, puis à partir d'un dictionnaire, la réponse la plus adaptée est déterminée et affichée. Pour assurer la qualité de ce chatbot, plusieurs conditions sont à assurer :

— la capacité du programme à correctement comprendre la question de l'utilisateur. Dans ce but, les phrases entrées sont divisées en ensembles linguistiques, eux-mêmes étiquetés puis pondérés. Cette pondération (cf. figure 1) est particulièrement importante, puisque c'est elle qui permet de sélectionner la réponse la plus appropriée parmi celles possibles ;
— un nombre conséquent de couples de questions-réponses doit être disponible. En effet, pour répondre selon le token indiquant la question, il faut grouper l'intégralité des couples de question-réponse selon le ce token. En pratique, pour une question précise, cela divise la taille du corpus de texte utilisé par un facteur six (voire plus) ;
— un accès rapide aux données. Il faut qu'une réponse ne mette pas trop de temps pour revenir vers l'utilisateur. Cela implique une gestion des données performante, dont le parcours est relaitvement limité dans le temps : c'est pourquoi un dictionnaire Python (concrètement, une table de hashage) a été utilisée. Les clés de cette table de hashage sont les mots interrogatifs (reconnus par *NLTK*), cela permet de diviser correctement les données.

Type de token	Poids	Exemples
Questions	2	"When", "What", "Who", "Where", "Why", "How"
Verbes	1.2	"Borrow", "Expand"
Noms	0.7	"John", "Doe"
Pronoms	0.7	"You", "Themself"
Adjectifs	0.25	"Awesome", "Magnificent"
Adpositions	0.25	"From", "With"
Adverbes	0.25	"Certainly", "Even"
Autres	0.05	" ?", "msitake", "666", ...

FIGURE 1 – Tableau de pondération des éléments syntaxiques

4.1 Les différentes pondérations

Il paraissait évident d'attribuer différent poids aux différents, mais pourquoi de tels choix de pondération ? Voici une explication selon les types de tokens identifiés par *NLTK* :

Questions : Détermine la nature de la réponse et sa tournure principale. C'est pourquoi la pondération de ce type d'éléments a été majorée ;

Verbes : Fixe le type d'action et permet généralement d'identifier l'action ou l'évènement concerné(e) par la question. Il permet de réduire correctement les réponses possibles : c'est pourquoi il s'agit de la seconde plus grande pondération ;

Noms : Un nom fait souvent référence à un contexte précis, ou tout simplement à un interlocuteur, parfois direct. C'est pourquoi il est important. Toufefois, si un nom n'est pas trouvé dans la base de données, il devient immédiatement sans importance. Cela devrait être fréquent, puisqu'il existe une infinité de noms propres. De plus, certains noms sont beaucoup plus fréquents que d'autres. Si une importance trop grande était accordée aux noms, alors une réponse en rapport avec ce nom, mais totalement hors-sujet pourrait être fournie ;

Pronoms : Les pronoms sont bien plus permissifs que les noms en termes de réponses. C'est pourquoi leur poids n'est pas non plus extrêmement important. De plus, ils sont utilisés dans beaucoup de contextes,

sans forcément avoir une importance majeure. Par exemple, lors d'une tournure rhétorique anglaise "..., don't you ?", l'information principale de la phrase est contenue dans la partie initiale de la phrase. Bien qu'une réponse courte du type "I do" soit valide, d'un point de vue conversationnel, il est plus intéressant de s'intéresser aux autres éléments de la phrase ;

Adjectifs : La majorité des adjectifs peuvent être remplacés par d'autres adjectifs sans forcément changer le sens de la phrase. C'est pourquoi ils ont un poids relativement faible. Néanmoins, la présence d'un champ lexical pour connaître les adjectifs permettrait d'élargir le champ de recherche des réponses vlaides en évitant en même temps les adjectifs qui changent effectivement le sens de la phrase ;

Adverbes : Les adverbes correspondent à un cas relativement similaire à celui des adjectifs, explicant leur poids faible ;

Adpositions : Les adpositions correspondent généralement à un complément d'information à l'action. Action qui reste l'un des éléments principaux de la phrase. Les adpositions ont donc une pondération faible ;

Autres : Les autres types de tokens comprennent les conjonctions, les déterminants, les nombres cardinaux, les particules, les mots issus d'une autre langue et enfin les fautes de frappes sont actuellement très peu importants à la réponse. En guise d'illustration, changer un nombre de valeur n'affecte que très peu le sens de la phrase (bien que les conséquences puissent être totalement différente selon le contexte). Toutefois, la pondération n'est pas nulle, juste très faible, car certains de ces mots pourraient revenir plusieurs fois dans une seule et même phrase, cela la mettrait en avant face à d'autres phrases similaires comportant moins de fois ces mots.

Les signes de ponctuation sont volontaires omis (pondération quasi-nulle) puisque l'entrée doit être similaire à une question pour que le programme trouve une réponse adaptée.

5 Améliorations possibles

5.1 Maintenir une analyse constante

Voici le scénario envisagé. L'utilisateur pose une question, le programme lui fournit une réponse. Une autre question est entrée, une nouvelle réplique est affichée. Et ainsi de suite. Pourquoi ne pas exploiter ce défilement d'informations potentiellement nouvelles ?

5.1.1 Enregistrer les entrées de l'utilisateur

Supposons que le programme échoue à analyser ou à répondre à l'utilisateur. Il pourrait demander à l'utilisateur de fournir une réponse convenable à sa question, et ainsi enregistrer un nouveau couple de question réponse.

Ou de manière bien plus transparente, il pourrait analyser la réaction de l'utilisateur suite à sa propre réponse. Il faudrait bien sûr une base de réponses désignant l'incompréhension, composée de phrases telles que "ce que tu dis n'a aucun sens" ou encore "tu racontes n'importe quoi". Si la machine reconnait une de ces phrases, elle stockerait une nouvelle association de question-réponse dans une table "à ne pas faire". Cette association doit avoir une valeur de fréquence, pour mesurer l'importance de cette réponse jugée inappropriée. En effet, certaines réponses de ce type pourraient relever du sarcasme (les smileys textuels ne sont absolument pas analysés).

5.1.2 Suivre un fil conducteur

L'idée est ici de mettre en place un historique et de l'analyser pour maintenir un lien concret entre une suite de question. Il pourrait être tout à fait pertinent que le programme se concentre sur certains thèmes ou champs lexicaux de la base de données. Par exemple, si la conversation est particulièrement orientée "sciences appliquées", il serait intéressant de favoriser les répliques provenant d'Interstellar et autres scripts issus de films ayant ce thème.

Concrètement, cela nécessiterait un travail de catégorisation des scripts, qui devrait se révéler relativement aisé, puisque les films sont généralement classés par genre. Mais comme les thèmes sont multiples et variés, d'autres méthodes sont certainement préférables.

5.2 Utilisation de champs lexicaux

Une de ces méthodes serait l'analyse de la question non pas seulement par sa probabilité de réponse selon un corpus de texte qui va finir par devenir volumineux, mais également par le champ lexical des mots de la phrase. En effet, connaître le champ lexical des noms communs, des adjectifs ou des verbes permettrait d'élargir le champ de recherche de réponse à cette question.

Cependant, cet élargissement du champ de recherche risquerait d'alourdir considérablement le temps de réponse, puisqu'il faudrait comparer toutes les phrases possibles pour obtenir la (ou l'une des) meilleure(s). Si ce temps d'attente reste tolérable, alors cette implémentation devrait être valide. A contrario, si la machine ne rétorque pas suffisamment vite, il faut envisager d'autres solutions.

L'une d'entre elles serait la génération de phrases. En pratique, le champ lexical n'est pas très long à récupérer si un système performant existe déjà. Le British Lexicon Project devrait a priori pouvoir aider sur ce point. Pour générer une phrase correcte, il suffirait de voir la structure de phrase la plus fréquente (à l'aide des tags *NLTK*), et de piocher dans le champ lexical des mots correspondant au type de token pour constituer une nouvelle phrase. Il s'agit bien sûr d'une méthode relativement basique ici, et d'autres plus élaborées existent certainement.

5.3 Correction orthographique

Une ultime amélioration apparente est celle de la correction orthographique des entrées de l'utilisateur. La manière transparente de faire serait d'accepter l'entrée de l'utilisateur, telle qu'elle est et d'y appliquer le correcteur orthographique dans l'espoir qu'il ne fasse pas d'erreur. Cette manière fonctionne particulièrement bien avec un unique (profil d') utilisateur, puisque via un processus d'apprentissage automatique, elle se familiarise avec le vocabulaire de l'individu (un des meilleurs exemples de ceci est le correcteur orthographique des smartphones).

De même, une solution valide serait de suggérer à l'utilisateur des corrections potentielles (via une interface graphique), qu'il sélectionnerait (l'illustration est ici le correcteur grammatical de Microsoft Office Word ou LibreOffice Writer).

Comment seraient suggérées ces corrections ? À l'aide d'un calcul de distance de Levenshtein, qui permet de calculer la similitude entre deux mots, en détectant les permutations, ajouts et suppressions de lettres pour arriver d'un mot à un autre. L'algorithme est relativement simple, mais s'il est appliqué à tout un dictionnaire, il risquerait d'être particulièrement gourmand en ressources. Limiter la corrections à trois ou quatre mots très proche semble raisonnable.

6 Exemple d'exécution

```
Did you love me ?
All the time .

Do you love me ?
Of course .

Where is my dog ?
She 's dead .

Where ?
In the shadow by those two trees .

Why ?
Do n't worry about it .

I should ?
What if your chute fails , Bodhi ?

Should I ?
I wrote you a letter right after I saw the article in the paper .

 What if we go ?
Where is everybody ?

Where is everybody ?
Well - it 's cold , Margie .

What are you doing ?
Cloud bursting .

Because of ?
Why should that bother them ?

Is that true ?
Have you told him I nearly drowned in the pond ?

Do n't you love me ?
Of course .

When ?
What is wrong with people ?

What 's that ?
Help , help , help , help

Where are you going ?
We have to get off this ship .

Do you speak ?
Of course I can , sir .

Yeah ?
( laughs ) Hey whadya say we take some of that Tijuana acid
and see what kinda trouble we can get into tonight ?
```