Rapport de Traitement Automatique des Langues – Worst-Friend Bot

MEUNIER Paul - LOCATELLI Pierre-Antoine

06/05/2017

1 Présentation du projet

1.1 Contexte

Durant ces dernières semaines, nous avons travaillé à la réalisation d'un projet en binôme dans le cadre de notre cours de Traitement Automatique des Langues. Il nous était demandé de concevoir un logiciel en Python, qui exploiterait diverses notions étudiées en cours, en TP, ou de manière autonome, telles que la tokenisation ou l'analyse sémantique de texte. Ce fut également l'occasion de se former au Python, un langage de programmation que nous avions peu utilisé jusque là, ainsi qu'à l'utilisation de GitHub pour gérer nos données, et à la rédaction d'un rapport en utilisant la langage LaTeX.

1.2 Objectif

Notre objectif initial était de permettre à l'utilisateur de converser avec un bot, que nous avons nommé Worst-Friend Bot, qui répondrait en anglais, et en utilisant principalement des citations ou des références à la culture plus ou moins populaire. Le but était également de lui donner un comportement antipathique, en lui fournissant un vocabulaire assez négatif, voire nonchalant, d'où son nom.

1.3 Déroulement du programme

Pour pouvoir utiliser le programme, il faut disposer d'un ordinateur doté de python 3, ainsi que de la bibliothèque spacy. Des indications pour l'installation de spacy sont indiquées dans le fichier ReadMe.txt. Exécutez le fichier WFB-Answer.py, attendez que le logiciel charge sa bibliothèque de mot, et le jeu peut commencer. L'utilisateur est invité à saisir une phrase, à laquelle l'ordinateur va répondre. Il pourra ensuite en proposer une autre, l'ordinateur répondra à son tour et ainsi de suite. Pour mettre fin au programme, écrivez seulement « stop ».

1.4 Contributions

Nous avons essayé de nous répartir les tâches le plus équitablement possible, de manière à ne pas nous gêner mutuellement, et à faciliter la fusion de nos travaux respectifs. Pierre-Antoine LOCATELLI s'est chargé de l'analyse des phrases proposées par l'utilisateur du logiciel, et de la séparation entre questions, affirmations et phrases incomprises. C'est également lui qui a implémenté le rapport en LaTeX, même si les deux élèves ont participé à sa rédaction. Paul MEUNIER quant à lui s'est occupé de toute la génération de texte plus ou moins aléatoire, et dépendante de la phrase entrée par l'utilisateur. Pour cela, il a entre autres rédigé, découpé et interprété entièrement le fichier DatabaseAnswer.txt. C'est aussi lui qui a fait la recherche des phrases cultes auxquelles l'ordinateur a accès pour formuler ses réponses.

2 Analyse de texte

Pour pour voir répondre à l'utilisateur, le Worst-Friend Bot doit d'abord comprendre ce que celui-ci lui dit. La première étape du programme consiste donc en l'analyse et le découpage des chaînes de caractères fournies par l'utilsateur, pour en déduire des informations sur son sens, et transmettre ces informations à la suite du programme, qui devra en déduire une réponse à afficher à l'écran. t.

2.1 Tokenisation et analyse grammaticale

2.1.1 Première approche

Le programme commence par la tokenisation de la phrase écrite par l'utilisateur, cet à dire de son découpage en mots ou en groupe de mots ayant le même sens. Au départ, nous utilisions un algorithme de découpage conçu lors du premier TP de TAL, adapté à la langue anglaise. Nous avons également essayé d'implémenter un algorithme associant à chaque token son type gramatical. Pour cela, nous avions un gros fichier servant de bilbiothèque de mots, qui associait ses nombreux mots de la langue anglaise à leur type grammatical, parmi entre autre « nom commun », « nom propre », « verbe », « adjectif », etc. Il suffisait donc de trouver les mots de la phrase proposée dans ce dictionnaire pour en connaître le type. On notera également qu'à chaque entrée de texte de la part de l'utilisateur, une version de la phrase où toutes les lettres majuscules sont converties en lettres minuscules est sauvegardée pour simplifier certains traitements.

2.1.2 Utilisation de spacy

Nous avons cependant été confronté à un problème au cours de notre progression, dû au fait que notre structure de données n'était pas très efficace. En effet, elle réunissait les mots dans un dictionnaire dont ils étaient les clés, sans prendre en compte le fait que le type grammatical d'un mot peut dépendre du contexte. Par exemple, en anglais « fly » peut être un nom ou un verbe selon le contexte, et notre méthode initiale ne prenait pas cela en compte. C'est une des raisons pour lesquelles nous avons décidé d'utiliser spacy, une bibliothèque de programmation Python destinée au traitement automatique des langues. En quelques lignes, cette bibliothèque charge son propre dictionnaire et permet de tokeniser un texte, tout en associant à chaque mot son type grammatical.

2.2 Choix du type de phrase

Nous avons décidé de répartir les phrases entrées par l'utilisateur en trois catégories :
— Les affirmations,
— Les questions,
— Les phrases incomprises,
Chaque phrase recevra par la suite un traitement différent selon la catégorie à laquelle elle appartient.

3 Génération de texte

Dans le Worst-Friend-Bot, la génération de réponse est programmée de plusieurs manières différentes. Chacune de ces implémentations pourrait être développé facilement avec le temps nécessaire. Nous avons privilégié la modularité du code, et il serait très simple de développer des nouveaux modules pour complexifier ceux en place.

Tout d'abord, nous disposons d'un fichier DatabaseAnswer.txt qui regroupe des phrases préconstruites, complètes ou contenant des balises sous forme de @MAJ dont la fonction est détaillée dans le haut du document. Le but est de remplacer ces balises par une expression qui a le sens ou la fonction grammaticale voulu dans la phrase. Une phrase ne peut avoir aucune balise @ST (@ST équivaut à @Something, prise à titre d'exemple). Les phrases préconstruites disposent d'un deuxième mécanisme qui va permettre de les utiliser dans le bon contexte : les tags de début de ligne (présents sous la forme <tag1—tag2—etc/>. Ces tags définissent un des sujets abordés dans la phrase et les mots clés associés. Par exemple :

<hello—hi—greetings—sup—what's up—yo/>What's up, buttercup

va être utilisée principalement pour dire bonjour et saluer l'utilisateur, si celui-ci le fait, car c'est lui qui commence par nous aborder. On voit que dans ce cas, si un des tags est trouvé dans une phrase de l'utilisateur, alors le

bot va systématiquement lui répondre une salutation, aléatoire parmi toutes les salutations disponibles. Il est donc possible que pour une même entrée, le chatbot renvoie la même chose, mais plus le corpus sera étoffé et complet, moins ce genre de cas sera possible. Il serait même envisageable d'enlever une phrase utilisée de la variable qui stocke la base de données des tags, et de recharger celle-ci quand le nombre de phrase disponible pour un tag donnée se rapproche de 0.

Parallèlement à cette génération, nous allons utiliser la bibliothèque Spacy pour faire du part-of-speech tagging et ainsi que de créer des relations entre les différents éléments de l'entrée utilisateur. Le part-of-speech tagging consiste à associer à chaque mot son rôle grammatical dans la phrase, par exemple nom, pronom, verbe, etc. En même temps, Spacy va créer un arbre de relations entre les mots de la phrase utilisateur qui va permettre de définir le sujet de la phrase, à quel verbe il est associé, quel est le complément d'objet direct par exemple. A partir de cet étiquetage de l'entrée utilisateur, on peut définir une méthode pour répondre à cette phrase en gardant un sens logique.

Nous partons donc du part-of-speech pour récupérer les mots clés de la phrase si jamais celle-ci n'a pas été identifiée comme fonctionnant avec les tags de la base de données privée. Spacy nous a énormément aidé pour nous permettre de remplacer facilement les tag @ST évoqués précédemment. En effet, cette bibliothèque défini un certain nombre de token évoqué précédemment qui permettent de construire facilement une phrase grammaticalement presque correcte qu'il est possible de récupérer à la volée et donc d'associer à une partie de phrase préconstruite à remplacer.

Enfin, à l'aide de tous les outils évoqués, nous allons pouvoir tenter de générer du texte de manière à répondre une phrase ayant du sens et ne copiant pas mot pour mot celle de l'utilisateur. Pour cela, nous allons utiliser une méthode très simple consistant à récupérer dans ce que l'utilisateur nous a fourni les tokens principaux : pronom, verbe, adjectif, et un nom (commun qui sera le sujet sur lequel la phrase va porter.

Dans cette dernière partie de génération, nous avons eu comme idée d'implémenter un pattern qui permettrait de générer une réponse correcte, mais en ayant dans l'idée qu'il serait facile d'en implémenter d'autre pourquoi pas via une base de donnée recensant plusieurs pattern ou en utilisant Spacy pour créer des pattern grammaticalement correct. Notre niveau de compétence nous a permis d'implémenter seulement un pattern très simple basés, comme dit précédemment sur un pronom, un nom, un verbe, et un adjectif optionnel. La génération se fait sous forme de question, et fait intervenir de manière aléatoire un effet de style de négation dans la question.

Pour finir, notre génération étant essentiellement basés sur des phrases écrites dans un fichier texte et dans le fichier de code, nous avons décidé, pour donner un peu plus de personnalité au bot, ainsi que pour le rendre plus vivant et lui donner un panel de répliques qui varie. C'est pour cela que lorsque cela est possible, le bot va lister toutes ses possibilités de réponses et en choisir une au hasard, pour éviter de ressortir toujours la même réplique. De petits ajouts viennent compléter le caractère énervant du bot, en lui donnant un petit panel de répliques qu'il donnera si l'utilisateur s'acharne à lui envoyer toujours la même entrée.

4 Limitations du programme

Le programme génère des réponses qui ont rapport avec l'entrée utilisateurs mais qui n'ont parfois aucun sens, soit dû au manque de données, l'utilisateur n'a entré qu'un mot, soit à cause du fait que la base de donnée n'est pas assez remplie.

Une des plus grosses limitations de ce programme est que toutes les entrées de DatabaseAnswer.txt ont été entrée à la main par les développeurs, ce qui en fait une contrainte majeure, surtout vu le peu de temps dont nous disposions pour l'achever. Il aurait fallu écrire un script Python permettant de récupérer des répliques, de films par exemples sur des bases de données du web, et tagger ces répliques avec des mots importants, comme par exemple ceux des répliques précédentes qui sont donc la meilleure source pour savoir ce qui doit suivre et permettre au bot d'avoir une personnalité propre. Enfin, pour l'instant, le programme est limité par sa base de données restreinte à un chat bot de type Eliza, qui se contente de répondre bêtement aux requêtes utilisateurs du mieux qu'il peut, certaines fois ne faisant aucun sens ou se répétant. Comme dit précédemment, agrandir la base de données aiderait, mais il faudrait implémenter un vrai algorithme de génération de texte basé sur des

corpus de texte, et pouvant même apprendre de ses discussions avec l'utilisateur.

5 Cas restants à traiter

Dans les cas restants à traiter, il ne nous reste pas grand-chose. La chose qu'il manque le plus à Worst-Friend-Bot est un sens de la conversation plus évolué. Son discours généré directement pas le programme sans ''aide" de la base de donnée est un peu limité et mériterait qu'on l'améliore, ce qui n'a pas été possible dans le cadre de ce projet. Cela serait possible en implémentant un lexicon permettant d'augmenter le vocabulaire disponible.

D'autre part, la bibliothèque Spacy est extrêmement riche et dispose d'outil très utiles. Malheureusement, leur documentation est un peu à la ramasse, et il est compliqué de trouver des solutions rapidement. Mais durant nos recherches, nous avons vu qu'il serait relativement simple d'implémenter un arbre de relations entre les mots plus complet, ainsi qu'une fonction de recherche d'entité avancée permettant de détecter les noms propres ainsi que ce à quoi ils sont associés (nom d'entreprise, de personnes célèbres, etc.) De plus, la bibliothèque est toujours en progression, ce qui veux dire qu'il sera sûrement plus simple dans le futur d'apporter des améliorations à ce bot si le besoin s'en fait sentir.