Mini Assistant Intelligent

Projet réalisé par Killian CHAMBELLANT, Nicolas DOUCHIN et Anaëlle MORIN à la demande de M. Nicolas DELESTRE dans le cadre des projets SOSI.

But du projet

Le but de ce projet est de créer un mini assistant capable de répondre à une question qui lui a été posée en langue naturelle. Dans un premier temps, cette question doit être reformulée par le programme pour devenir une requête formelle permettant d'interrogée la base de données. La réponse fournie doit être récupérée par le programme dans un type de base de données particulière du web appelée le web des données liées.

Ce genre de programme a déjà été traité par le passé. Il nous a donc été demandé d'utiliser le framework Python Quepy qui permet déjà de transformer une question posée en langage naturelle en une requête formelle. À partir de cela trois prototypes étaient à fournir pour explorer différentes possibilités :

— La première version devait reprendre le travail réalisé par Machinalis et fonctionné comme le site `http://quepy.machinalis.com/`. On avait ici une question en anglais qui était traitée et la réponse cherchée dans la base DBPedia.
— La deuxième version devait permettre de poser les questions en français au lieu de l'anglais.
— Enfin la troisième version devait requêter la base de données Wikidata.

Il est à noter que le développement de Quepy a été arrêté depuis deux ans.

Protoype 1

Comme indiqué ci-dessus, l'objectif de ce premier prototype était de prendre en main la librairie Quepy afin de recréer une application capable de répondre aux mêmes questions que celles montrées en exemple sur le site de Quepy. Il a donc fallu comprendre toute la chaîne de traitement derrière l'application. Celle-ci était expliquée grâce au tutoriel proposé sur le site de l'application, il permettait de construire étape par étape une application utilisant Quepy et d'implémenter différents types de questions. Pour les implémenter, il a fallu comprendre le fonctionnement de Quepy à ce niveau, nous avons pu voir que celui-ci était très proche de ce que nous avions vu au niveau de la compilation. Quepy utilise un tagger qui est un outil linguistique permettant de réaliser une analyse lexicale sur un langage naturel, Quepy utilise NLTK. Le tagger comprend les notions de tokenizer, d'étiquetage morpho-syntaxique et de lemmatisation. Le tokenizer va se charger de délimiter les éléments d'une chaîne de caractères écrite en langage naturelle et s'occupera éventuellement de les classer. A partir du résultat fourni par le tokenizer, l'étiquetage morpho-syntaxique va pouvoir associer aux mots du texte des informations grammaticales correspondantes. Par exemple, le mot « nous » sera indiqué comme pronom et « sommes » comme verbe. La lemmatisation a pour rôle de regrouper les mots d'une même famille. Par exemple, on pourra regrouper les différentes formes d'un verbe : Lemma("be") regroupera aussi bien le verbe à l'infinitif de be, que sa forme au présent ou encore au passé. A partir de ces outils,

il sera alors possible de définir des expressions régulières permettant de définir les différents types de questions. Par exemple, pour la question « Who is Tom Cruise ? », la formation de l'expression régulière est réalisée de cette manière : Lemma("who") + Lemma("be") + Person() + Question(Pos(".")). On va utiliser la lemmatisation pour le verbe « be » afin de gérer les différents temps possibles. Et « Person() » sera une classe qui va s'assurer de récupérer le nom de la personne recherchée. Grâce à ce modèle, nous avons pu implémenter une nouvelle question (en plus de celles déjà proposées de base) qui est « Birth/Death date of Tom Cruise ? » et qui se base sur les principes vus précédemment. L'information demandée est récupérée dans la base de données wikidata comme prévu par le fonctionnement de base de l'application.

Protoype 2

Le but de ce prototype est de reprendre le premier afin de pouvoir poser des questions en français.

Pour pouvoir traiter une question en français il faut modifier le tagger qui gère le passage des questions en anglais (langage naturel) vers un langage formel. Quepy utilise nativement nltkdata. Nous avons regardé s'il était possible d'utiliser un autre tagger qui aurait pu être soit une version française de nltk soit un autre. Malheureusement, après analyse du code de quepy, nous en sommes venus à la conclusion que quepy était intrinsèquement lié à nltk (il y a un module python qui s'occupe de la liaison avec nltk qui est directement inclus dans quepy). L'objectif du projet ne prenait pas en compte la modification de quepy. Et concernant nltk français, pour configurer quepy il aurait également fallu modifier quepy. Pour réaliser le prototype 2, nous avons décidé d'utiliser une solution tierce non viable sur le long terme qui est de traduire avec google traduction les questions posées par l'utilisateur puis d'envoyer le résultat à quepy. Comme dit précédemment, cette solution n'est pas viable sur le long terme car nous pourrions avoir des erreurs de traductions sur des questions plus complexes. À noter également que google traduction gère correctement le passage du français vers l'anglais, mais que cela n'est pas vrai pour toutes les langues. À terme, il sera donc nécessaire d'examiner une solution via un tagger pour gérer le support linguistique du programme.

Prototype 3

Le principe de ce prototype est de reprendre le programme et de l'adapter afin d'interroger la base de données "Wikidata".

Wikidata est une base de données linguistiquement neutre et est basée sur des faits et non pas sur des opinions comme DBPedia qui est alimentée par Wikipédia.

Le choix d'implémentation de base de quepy fait qu'il renvoie une chaine de caractères et non pas l'URI. Alors que nous voulions récupérer l'URI afin de pouvoir réaliser des requêtes sur des éléments précis et de pouvoir moduler les réponses. Pour pouvoir récupérer l'URI nous avons été obligés de modifier la requête générée à la volée avant qu'elle soit envoyée à Wikidata. Grâce à l'URI récupérée et aux metadatas nous générons d'autres requêtes pour obtenir les informations pertinentes recherchées. Pour cela, nous avons ajouté un package contenant des modules permettant facilement l'ajout de types de question. Notamment par la déclaration de méthodes selon la metadata obtenue.

Pistes d'amélioration

Notre solution, bien que fonctionnelle pour quelques questions, présente différents défauts. Tout d'abord, quelque soit le prototype, le nombre de questions traité n'est pas très élevé. De plus, comme le prouve l'amélioration faite dans le prototype 1, il existe des questions traitées mais qui peuvent encore être améliorées.

Les regex ne sont pas parfaites non plus. Par exemple, un prénom contenant un "-" ne sera pas pris en compte ou la question pour "What is" pour une entité de plusieurs mots telle que "Institut national des sciences appliquées de Rouen" ne fonctionne pas encore. Mais grâce au découpage du code ce sont des améliorations faciles à faie.

Une autre chose à améliorer aurait été de permettre dès le prototype 1 l'envoi de plusieurs requêtes pour une seule question formulée. En effet, actuellement l'âge retourné pour une personne morte est le nombre d'années écoulées depuis sa naissance car, de part sa découpe, le code ne permet pas d'envoyer une requête à la fois pour la date de naissance et pour celle de mort. Une plus grande modularité est donc requise comme faite à partir du deuxième prototype.

Enfin, comme dit précédemment, la gestion de la langue française en passant par google traduction n'est pas viable sur le long terme et ne permet pas la gestion de questions trop compliquées. Il reste à trouver et à faire fonctionner un tagger.