Université de Namur

IHDCM036 - Techniques d'Intelligence Artificielle

Projet : Construction d'un bot - conseil en vins

Groupe 1

Etudiants :
Thomas Wery
Alexis Degraeve

Professeur :
J.-M. Jacquet

01 Janvier, 2022

Table des matières

Chapitre 1

Introduction

Dans le cadre du cours de "techniques d'intelligence artificielle", nous avons réalisé un outil de conversation basé sur le langage Prolog. Ce chatbot a pour but de répondre aux questions de clients concernant des vins que la société "Domaine du bon vin" met en vente.

L'objectif de ce document est de décrire notre approche et la manière dont nous avons réaliser la tâche demandée. Il s'agit, dès lors, d'un programme se limitant aux caractéristiques des directives de ce travail et qui se veut orienté vers le monde viticole. Notre travail est donc une ébauche de bot conversationnel car le temps nécessaire à la réalisation d'un programme complet dépasserait le délais imparti pour ce cours.

Nous avons prêté aussi également un soin particulier à l'interface pour qu'elle soit conviviale et simple d'utilisation.

Chapitre 2

Objectifs du projet

Dans le cadre de ce projet, notre programme possède une base de connaissance composé de différents faits et règles permettant de lire et de formuler une réponse aux questions de l'utilisateur.

Pour ce faire, nous avons implémenté notre système de chatbot dans une interface web basé sur une combination de Prolog, HTML, CSS et JavaScript. Cela inclut également l'utilisation de Tau-Prolog pour pouvoir interpréter le prolog en Javascript.

C'est JavaScript qui incorpore les fonctionnalités d'intelligence du chatbot et ce, grâce à l'utilitaire "tau-prolog". Voir sur le site Tau Prolog, pour plus d'information sur la librairie.

Nous avons également fait appel à un moteur de reconnaissance vocal pouvant remplacer la saisie manuelle de texte.

Ce moteur de reconnaissance vocal est WebSpeech dont l'utilisation est expliqué sur MSDN, pour savoir comment l'utiliser, voir sur le lien suivant Utiliser l'API Web Speech

Chapitre 3

Environnement du projet

Comme expliqué précédemment, afin de réaliser une application plus interactive, nous avons décidé d'intégrer notre AI dans une page web écrite au moyen de HTML , CSS et JavaScript. Le tout appuyé par "tau-prolog".
Le design de l'interface a été conçu avec simplicité et élégance afin de refléter le niveau de qualité et de sérieux de la société cliente.
Nous avons également exploiter la librairie speech (en JavaScript) permettant à l'utilisateur d'interagir vocalement avec le logiciel et de rendre l'expérience client plus authentique.

Nous avons utilisé le **singleton design pattern** pour instancier la session du Tau Prolog. Nous pouvons ouvrir la session de chat soit par le bouton dans la page d'accueil mais également par le bouton dans le menu du haut. Comme il y a deux points d'accès, il est inutile d'instancier plusieurs fois la même session tauprolog, ainsi l'utilisateur pourra continuer la discussion où il en était avec le bot.

Afin que les informations circules de manière fluide entre l'interface et la session Prolog, nous utilisons des techniques de requêtes asynchrones envoyant les requêtes de l'utilisateur sous forme de vecteurs numériques.
Ensuite, ces requêtes sont traitée par le programme Prolog. Celui-ci se base sur une série de règles associées à des mots clés permettant de fournir la réponse la plus appropriée aux désirs de l'utilisateur. Dans le cas où une question est mal formulée ou se réfèrent à des éléments inconnus des faits du programme Prolog, une réponse adéquate est formulée spécifiant la non compréhension ou la non présence d'un élément.

Concernant l'implémentation de prolog dans une page web, l'utilisation de "tau-prolog" (dans sa version 0.3.1) nous a poussé à certains aménagements.
En effet, nous avons dû créer plusieurs fonction en JavaScript permettant la conversion d'une séquence de caractères en un array de nombre.
Nous avons également du adapter notre interface afin que la réponse reçue dans la méthode "answer" puisse être correctement manipulée par la session prolog.
Enfin, nous avons également du apporter un soin particulier au format d'affichage de aux réponses renvoyées par la session prolog afin que celles-ci soit rédigée dans une forme et un synthaxe proche du langage naturel (espace entre les mots, ...) .

Une fois la réponse transmise et formatée, celle-ci est transmise dynamiquement au innerHTML afin qu'elle s'affiche à l'utilisateur.

Notre interface s'inspire grandement des systèmes de messageries les plus communs (Messenger, Whatsapp,...) afin de rendre le système plus intuitif pour l'utilisateur.
Les points d'inspirations ont notamment été l'enchaînement des questions-réponses, le visuel et le design utilisé.

Quand on entre une question dans le chat et on presse enter, l'avatar du bot s'anime et change de couleur. Elle change en jaune s'il peut répondre à la question et aussi non en rouge dans le cas contraire.

Au sein de la session Prolog, nous avons choisi de nous baser sur des règles simples permettant de répondre aux questions énoncées dans le descriptif du projet.
Nous avons également intégré des variations de phrases ou de termes spécifiques à la langue française afin de rendre le programme plus souple à l'utilisation.

Nous avons également remplacer les caractères spéciaux de la langue française en JavaScript par des caractères non accentués pour que le PROLOG puisse mieux les interpréter car au final cela signifie la même chose. Nous avons utilisé **un regex** en JavaScript pour permettre le remplacement des caractères.

La base de connaissance est inspirée des vins présents sur le site de Grafé-Lecocq et reprends une sélection d'environ 20 vins de différentes sortes.

Par les connaissances d'Alexis (designer de profession), nous avons réaliser une interface simple et conviviale.
L'utilisateur se connectant à notre chatbot se retrouve sur une page d'accueil lui permettant d'engager la conversation avec Gus (notre Chatbot). Gus se veut empathique dans la manière de s'adresser à l'utilisateur afin de pouvoir aider au mieux un potentiel client.

Chapitre 4

Fonctionnement du programme

Dans cette section, nous détaillerons le fonctionnement de notre programme et la séquence que celui-ci suit lorsqu'un utilisateur lui pose une question.
Nous détaillerons également les règles et faits qui permettent à Gus de répondre aux questions posées.
Afin que le programme déclaratif fonctionne correctement, nous nous sommes basé sur des métarègles.
Celles-ci nous permettent de pouvoir appeler de manière multiple des foncteurs en redéfinissant leurs paramètres.
En effet, nous ne pouvons pas invoquer les foncteurs et des paramètres directement et les métarègles nous permettent de solutionner ce problème.
Afin de pouvoir unifier le "head" et le "body" au travers du foncteur "clause/2", nous avons du rendre les prédicats public. Ceci est possible grâce à la fonction " :-dynamic(regle$_r ep$/4)"

La lecture de la question d'un utilisateur se déroule comme suit. Notre programme parcourt un foncteur récursivement. Ce foncteur contient comme premier argument un tableau. Ce tableau est composé de deux éléments, une tête de lecture ("H") et une terminaison ("T"). Grâce à ces deux éléments, nous pouvons extraire successivement chaque élément jusqu'à ce que le tableau soit vide.
En voici un exemple, la phrase "A Noël j'envisage de cuisiner du canard quel vin me conseillez-vous ?". Nous avons trois mots clefs importants : "Noël", le mot clef "canard" qui est un type de viande et le mot clef "vin".
À partir de ces deux mots clef, nous pouvons transmettre le type de viande vers un foncteur qui va nous retourner le ou les types de vins appropriés. Une recherche est ensuite effectuée dans la base de connaissance pour y trouver les vins correspondant au plat. Dans le cas contraire, si aucun vin n'a été trouvé, un message est généré. Celui-ci indique à l'utilisateur qu'aucun vin ne correspond au plat.

Cependant, cette manière de procéder nous a confronté à un soucis de mémoire. En effet, les chaînes de caractères nécessitant un espace mémoire plus important, nous avons du augmenter la mémoire allouée.
En nous plongeant dans la documentation de "Tau-Prolog", nous avons constaté que l'argument passé en paramètre de la fonction "create" permet d'allouer un certain volume de mémoire à l'application. Nous avons alors adapté ce volume afin que les erreurs générées dans la console ne surviennent plus.

```
1 class PrologSession {
2   session = pl.create(40000);
3   ...
4   }
```

Listing 4.1 – class PrologSession

Afin de rendre la saisie et la lecture du texte plus naturelle, nous utilisons la fonction "nom_vins_uniforme" qui établit une corrélation entre un identifiant fournit pour un vin et le nom qui sera affiché et/ou utilisé par le client.

Le remplacement s'effectue en liant l'identifiant d'un vin au sein d'une fonction ("*beaumes_de_venise_*2015") avec un nom compréhensible et utilisable ("Beaumes de Venise 2015").
Une fois ce lien fait, on chaîne les différentes associations (L1-L2, L2-L3, L3-L4,...) dans le foncteur "*nom_vins_uniforme*". Le noeud terminal est ensuite affecté à la sortie ("*L_mots_unif*").
Les noms sont ainsi liés à leur identifiant.

```
nom_vins_uniforme(Lmots,L_mots_unif) :-
   L1 = Lmots,
   replace_vin([beaumes,de,venise,2015],beaumes_de_venise_2015,L1,L2),
   replace_vin([chateau,saint,simeon,2014],st_simeon_2014,L2,L3),
   replace_vin([pommard,la,chaniere,2017],pommard_la_chaniere_2017,L3,L4),
   replace_vin([ventoux,aoc,2019],ventoux_aoc_2019,L4,L5),
   replace_vin([beaune,premier,cru,2018],beaune_premier_cru_2018,L5,L6),
   replace_vin([nuit,saint,georges],nuit_saint_georges,L6,L7),
   ...
   L_mots_unif = L21.
```

Listing 4.2 – Associer un nom de vin à un id unique

Afin de pouvoir réaliser une recherche dans notre base de connaissance, nous avons dû utiliser le foncteur "findall".
Celui-ci permet de retourner tous les faits qui répondent aux critères choisis.
Par exemple, si nous souhaitons savoir quels vins s'assemblent avec un plat à base de canard, le programme nous retournera cette sélection dans la variable "AppelConseil". Le détail de cette manipulation est présent ci-dessous.

```
regle_rep(noel,12,
  [a, noel,jenvisage,de,cuisiner,du,Repas,quel,vin,me,conseillezvous],
     Rep ) :-
     findall(X, platvin(Repas, X), Appelconseil ),
     rep_appelvin(Appelconseil, Rep).
```

Listing 4.3 – Répondre à une requête

Une fois la liste des vins correspondants reçues, nous procédons à sa mise en forme. Ceci afin que la réponse lue par l'utilisateur soit présentée au moyen d'une interface agréable et non sous la forme d'une liste brute.

Nous avons également implémenté le choix aléatoire de certaines réponses à quelques questions spécifiques qu'un utilisateur pourrait poser. Si l'utilisateur entre le mot "Bonjour", une réponse choisie aléatoirement parmi une liste est affichée.
Pour se faire, nous utilisons le foncteur "Random_member". Celui-ci choisira aléatoirement une réponse parmi un tableau contenant les réponses possibles. La réponse sélectionnée est ensuite retournée via la variable "Rep" et affichée (voir ci-après).
Cette fonction n'est utilisable qu'en appelant la librairie "random".

```
:- use_module(library(random)).

produire_reponse([bonjour], Rep) :-
   random_member(Bonjour, ['Bonjour Cher visiteur','Salut visiteur','Hello visiteur']),
   Rep = [[Bonjour]].
```

Listing 4.4 – réponse aléatoire

Gus est ainsi capable de répondre à série de questions tel que :

- Bonjour, Salut, Coucou
- Ça va ?, Comment allez-vous ?.
- Merci
- Au revoir, À bientôt, etc.
- Que donne le **** en bouche ?
- Auriez-vous des vins entre *** et *** € ?
- Pourriez-vous m'en dire plus sur *** ?
- Quel nez présente le *** ?
- À Noël, j'envisage de cuisiner du ***, quel vin me conseillez-vous ?
- Que recouvre l'appellation *** ?
- Auriez-vous un *** ?
- Combien coûte le *** ?
- Quels vins de *** me conseillez-vous ?
- Auriez-vous d'autres vins de *** ?

Il dispose également d'une base de connaissance de vins basé sur les critères suivant :

- Nom
- Prix
- Couleur
- Description
- Nez
- Bouche
- Appellation

Chaque vin est également lié à une région et un ou plusieurs plats(s).

Il peut donc fournir suffisamment de renseignements à l'utilisateur sur les caractéristique, associations et origines des vins disponible dans le catalogue de la société "Domaine du Bon Vin".

Conclusion

Ce projet nous aura permis de découvrir et d'apprendre les caractéristiques et les spécificités d'un langage déclaratif.
Nous avons dû adapter notre manière de penser la conception d'un logiciel pour que celle-ci puisse correspondre au fonctionnement d'un langage tel que Prolog.
Nous avons également appris à intégrer un tel langage au sein d'une architecture web. Ceci grâce à différents outils (Tau-Prolog, Speech). Ce projet nous a confronté à pas mal de défis afin de pouvoir remplir les objectifs fixés et nous a obligé à changer notre manière de travailler et a fortement collaboré afin que les forces de chacun puissent être exploitée. Nous avons également appris un grand nombre de choses sur le monde viticole.
Tout ceci dans le but de fournir un chatbot répondant aux objectifs de manière simple et efficace et ce, au moyen d'une interface agréable pour l'utilisateur.

Avec plus de temps, il serait tout à fait possible d'améliorer Gus en enrichissant sa base de connaissance.
Nous pourrions également déplacer cette base de connaissance vers une base de donnée externe au programme Prolog utilisant un langage dédié à la manipulation de données (SQL, ...).
Nous pourrions également élargir le domaine de connaissance de Gus à autre chose que les vins (digestifs, cocktails, ...) et y ajouter d'autres évènements (mariages, soirées, etc.).

Pour conclure, nous pourrions simplement préciser que nous avons pris beaucoup de plaisir à réaliser ce projet. Le langage déclaratif était quelque chose qui nous était complètement inconnu jusqu'à ce jour et le cours de T.I.A nous a permis d'élargir notre connaissance et d'apprendre d'autres manières de réfléchir et de programmer.