StrASKbourg

Benjamin Brindamour - Marie Cheng - Pierre Genty

6 mai 2017

1 Motivation et objectif

StrASKbourg est un chatbot de recherche sur la ville de Strasbourg. Le but est de renseigner à l'utilisateur les informations qu'il demande en formulant des phrases. Par exemple, si l'utilisateur demande "*Puis je aller au macdo ?*", on s'attend à ce que le bot réponde quelque chose comme"*Oui, le plus proche est ....*". On veut aussi qu'il puisse comprendre des approximations familières comme dans l'exemple précédent, avec *macdo* à la place de *Mc Donald's*, ou des ordres plus directs comme "*Liste moi les lieux touristiques et leurs adresses.*"

Notre choix s'est porté sur la ville de Strasbourg non seulement par affection mais aussi par praticité : la ville met à disposition des bases de données très complètes pour son projet *Open Data*, l'ouverture de ses données au grand public pour se lancer dans le numérique. Pour en savoir plus : `http://www.strasbourg.eu/web/strasbourg.eu/ma-situation/professionnel/open-data` . Toutefois, malgré notre intérêt pour la ville, ne la connaissant pas il nous est impossible de vérifier la véracité des informations fournies

Le bot se limite à la recherche d'informations sur les lieux de Strasbourg, et plus particulièremnt dans un but de divertissement dans la ville. Le projet ne gère donc pas d'autres villes, ainsi que des recherches d'informations allant au delà des frontières de Strasboug. Le bot ne donne aucun avis non-objectif (basé sur des évaluations par exemple) et se fie uniquement aux informations qui le composent, par exemple il ne peut choisir "le meilleur" de deux restaurants sans utiliser une base extérieur.

2 Le bot

2.1 Mécanismes

Le bot est réalisé en Python 3. La base de données que nous utilisons est un ensemble de terme que nous avons défini permettant de repérer avec précision les volontés de l'utilisateur. Pour une discussion plus naturelle et une recherche mieux guidée, le bot a une mémoire sous forme d'une pile, notamment pour comprendre à quoi se réfèrent les pronoms appelés par l'utilisateur.

2.2 L'initialisation du bot

Le programme est constitué de deux classes principales, gérant respectivement les élements textes et le déroulé des demandes, ainsi que de trois fonctions globales permettant la mise en forme du texte au moment de l'initialisation.

2.3 Décomposition du texte

Une chaine de caractère n'est pas lisible pour le bot, et il est donc nécessaire de l'enrichir pour la rendre compréhensible par la machine. Ainsi, Tout élement est inséré dans une classe ElementTexte, dont l'initialisation fait appel à nos fonction globales, qui dans un premier temps vont éclaicir le texte en retirant les élements superflus et en les mettant au même niveau (par exemple, seulement un seul type d'apostrophe) via la fonction normalise. On reppart ensuite de cette base afin de découper le texte en élements fondamentaux, le programme pouvant comprendre les information mot à mot. On utilise pour ça la fonction tokenise. Toutefois, malgré le

découpage, ces informations restent vide de sens aux yeux du bot. Pour lui permettre de finalement comprendre la demande, on ajoute deux couches d'information, donnant dans un premier temps le lemme du mot (on associe de cette manière les mots aillant le même sens), puis ensuite sa signification, nous permettant ici de déduire le but de la demande via les lemmes vu précédement.

2.4 Exécution

Une fois l'ensemble des éléments textes transformés, il devient plus simple d'éxécuter le programme, en utilisant cette compréhension des phrases afin de saisir le sens des saisies utilisateurs

3 Difficulté

Dans la conception, plusieurs problèmes majeurs se sont présentés à nous

3.1 Le bruit

Dans la conception d'une application liée à la reconnaissance d'une saisie humaine, la plus grande complexité a été le gestion de se paramètre variable qui ne peut être compris que partiellement dans un algorithme. Ainsi, un utilisateur trouvera toujours un moyen de faire une saisie incompréhensible, malgré la pensée que tous les cas étaient couvert. Pour résoudre, du moins partiellement ce problème, nous avons analyser l'origine des erreurs les plus courante. Une des plus commune est l'oubli des accents dans les mots, mais d'autre sont bien plus dur a saisir, comme le langage SMS, qui nécessite une compréhension quasi phonétique du texte. Pour cela, en plus de notre normalisation, nous avons pris en compte plusieurs orthographes d'un même mot, parfois légèrement différentes de la réalité, afin de couvrir plus de cas réel. De plus, nous avons ajouter une fonction permettant le remplacement de certain mot de langage SMS courant par les véritables expressions françaises.

3.2 Technique

Le premier blocage technique a été un barrière entre l'encodage sous linux et sous windows, nous empêchant de récupérer notre travail de TP. Enfin, le second est bien plus sociétal, est lié à la difficulté à se procurer des données en France. Malgré la certaine ouverture de la ville de Strasbourg, les données restent dures à utilisée, ainsi qu'à analyser

3.3 Exemples d'utilisation

4 Problèmes rencontrés et bugs connus

Le bot est conçu pour reconnaitre les élements fondamentaux d'une phrase pour ensuite les passer dans la table des désirs, lui permettant ensuite de comprendre les besoins. Par exemple, dans la phrase "Je veux manger chinois", veux montre la volonté, manger, le but, et chinois, le lieu. Cela peut être décliné sur tous les centres d'interêt majeur

4.1 Problèmes rencontrés et leur résolution

Avec les problèmes d'encodage et notre apprentissage récent du python, nous n'avons pas pu nous connecter et créer nos base de données. Nous avons donc créer une base nous même en définissant les mots importants et significatifs.

4.2 Bugs connus

5 Idées d'amélioration

Il existe une base de données en temps réel des transports de Strasbourg. Nous pourrions l'utiliser dans notre système pour avoir une palette de recherche plus riche.

Pour que notre bot soit efficace, il serait judicieux qu'il ait une interface graphique et qu'il soit portable sur mobile et tablette. En ayant la position en temps réel de l'utilisateur, il pourrait aussi lui fournir des itinéraires en le liant avec *Google Maps*.

Pour aller encore plus loin et suivre au mieux le dynamisme de l'utilisateur, on pourrait envisager d'ajouter de la reconnaissance et de la synthèse vocale, ainsi que l'intégration des bases de l'état