



ORACLE, L'ASSISTANTE 2.0

Rapport de projet

Résumé

Au cours du troisième semestre de mon DUT Informatique, j'ai eu l'occasion de participer, avec mon groupe de projet et sous la tutelle du professeur Christophe Nicolle, à l'élaboration d'un agent conversationnel (chatbot en anglais), un assistant artificiel capable de comprendre le langage naturel d'un utilisateur, de traiter ses requêtes et de lui répondre avec une formulation correcte.

Dans la prolongation de ce travail, j'ai décidé au cours de ce semestre de pousser encore plus loin ce concept en créant Oracle, la version 2.0 du Checkbot (chatbot créé pour CheckSem, le laboratoire de M. Nicolle).

Alors que le but lors du dernier semestre était de développer un outil, l'objectif cette fois-ci était de créer une assistante.

Alexis Guyot

IQS4-B2

TABLE DES MATIERES

PREMIÈRE PARTIE : PRÉSENTATION DU PROJET	4
I. INTRODUCTION	4
II. PRÉSENTATION DE L'EXISTANT	5
III. PRÉSENTATION DES OBJECTIFS	6
DEUXIÈME PARTIE : TRAVAIL EFFECTUÉ	8
I. ORGANISATION	8
II. ACHAT ET MISE EN PLACE D'UN SERVEUR	8
1. LE CHOIX DU SERVEUR	8
2. DE RASA À LUIS : CHANGEMENT DE CERVEAU	8
III. ORACLE, PARLE-MOI ...	9
1. LA VOIX DU SAVOIR	9
2. LA RECONNAISSANCE VOCALE	10
IV. ORACLE, QUEL TEMPS FAIT-IL ?	11
1. RÉCUPÉRER LA MÉTÉO	11
2. LOCALISER L'UTILISATEUR	12
V. ORACLE, SOURCE DE SAVOIR	13
1. QUI EST CETTE PERSONNE ?	13
2. CONNAIS-TU CETTE VILLE ?	14
VI. AMÉLIORATIONS MINEURES	15
1. MODIFICATION DE L'INTERFACE	15
2. MISE EN PLACE D'UN HISTORIQUE	16
TROISIÈME PARTIE : POUR CONCLURE ...	17
I. COMPETENCY QUESTIONS - RÉSULTATS	17
II. BILAN DES COMPÉTENCES ACQUISES	18
III. PROLONGATIONS POSSIBLES	19
ANNEXES	20
I. TESTER ET UTILISER ORACLE	20
II. PLANNING D'ORGANISATION	20

PREMIÈRE PARTIE : PRÉSENTATION DU PROJET

I. INTRODUCTION

Avant de commencer à parler des spécificités, il est important de répondre à une question de base : Qu'est-ce qu'un chatbot ? Le site [futura-sciences](https://futura-sciences.com) propose la définition suivante :

Un chatbot, aussi appelé « agent conversationnel », est un programme informatique capable de simuler une conversation avec un ou plusieurs humains par échange vocal ou textuel.

De cette définition, on peut retenir un élément caractéristique d'un agent conversationnel : il est capable d'avoir une conversation avec un être humain. Et pour ce faire, celui-ci doit être capable d'acquérir la requête de l'utilisateur dans son langage naturel (le français, l'anglais, tous les langages parlés par les êtres humains), il doit la comprendre, c'est-à-dire déterminer les intentions de la personne en face, trouver ce qu'elle veut, et enfin lui répondre dans ce même langage. Un programme informatique capable d'effectuer ces actions est un chatbot.

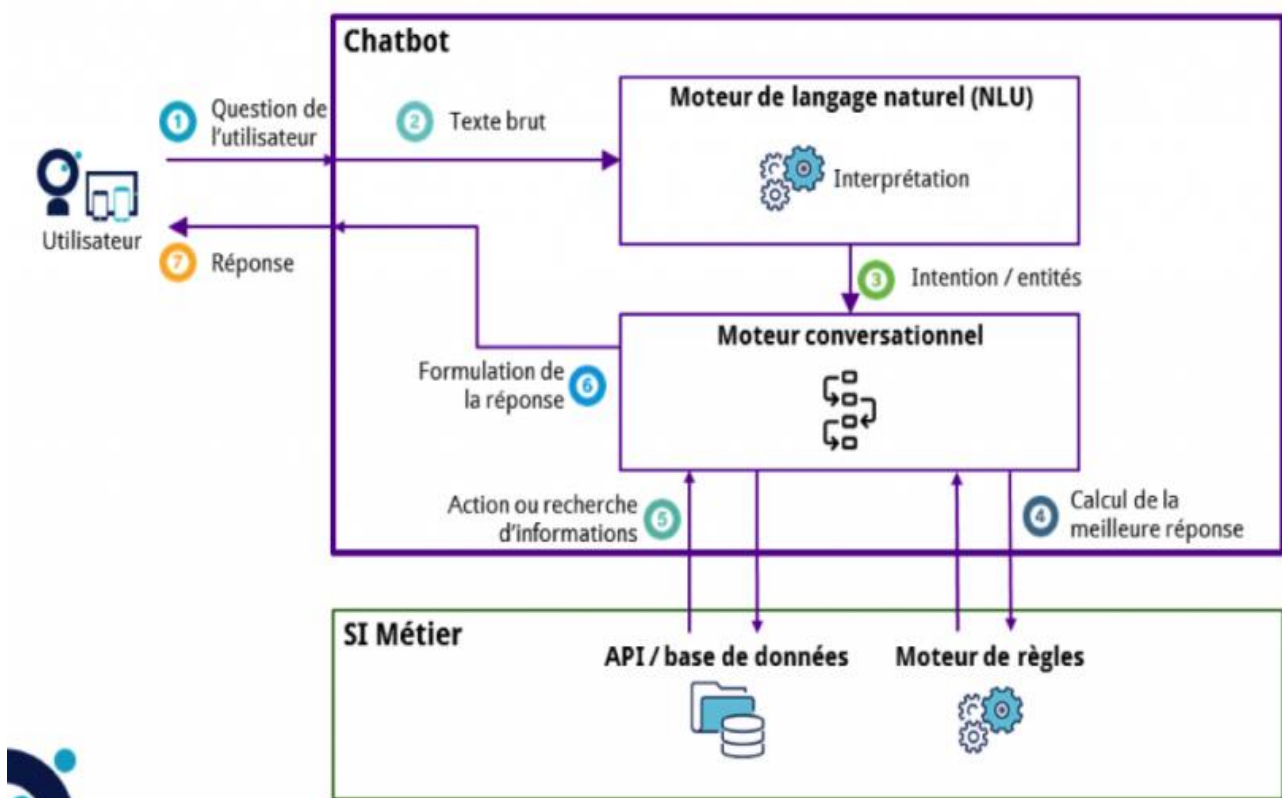


Figure 1 - Représentation schématique du fonctionnement d'un chatbot

Un chatbot est également défini par son niveau d'intelligence. L'évolution des technologies a permis d'améliorer considérablement la façon dont les agents conversationnels comprennent le langage humain. Eliza par exemple, le premier chatbot créé en 1966 par le MIT¹, se basait sur un ensemble de questions prédéfinies, représentées par des mots-clés. Quand le robot repérait un mot-clé qu'il connaissait, il cherchait

¹ Massachusetts Institute of Technology

la réponse associée et la renvoyait. S'il ne trouvait rien, il se contentait de reformuler la phrase de l'utilisateur. 50 ans plus tard, l'avènement de l'intelligence artificielle a permis de considérablement faire évoluer le domaine, et aujourd'hui les algorithmes de traitement du langage naturel (NLU sur la figure 1) se basent sur des technologies de l'IA. Ainsi, comprendre une phrase aujourd'hui ne veut plus dire associer un mot-clé à une réponse, mais veut dire identifier une intention derrière un message, et éventuellement quelques entités, qui dans notre langage représentent le sujet de la requête (Par exemple dans la phrase « Quel temps fait-il à Paris ? », l'intention est de récupérer la météo, l'entité est « Paris »).

Les chatbots prennent aujourd'hui une grande importance dans le monde professionnel, puisque les plus performants sont totalement capables de remplacer des assistants humains. De plus en plus d'entreprises cherchent donc à s'en procurer un pour assurer leurs services avant et après-vente et leur service d'assistance en ligne (ou téléphonique).

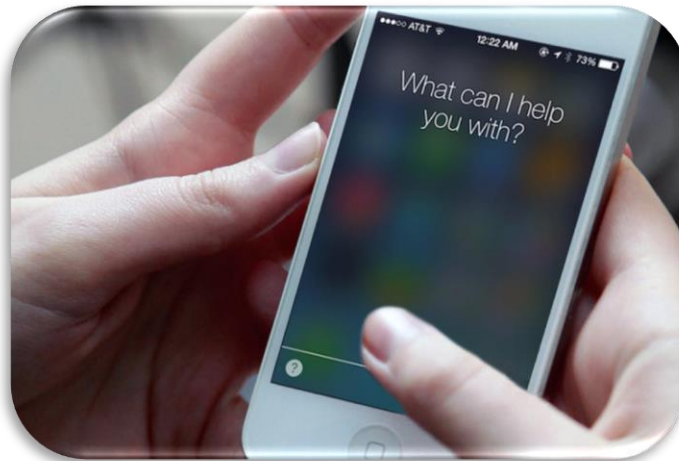


Figure 2 - Siri, créé par Apple, est l'un des chatbots les plus performants du monde

II. PRÉSENTATION DE L'EXISTANT

Dans ce contexte-là, et parce qu'il allait avoir besoin d'une base pour créer quelques chatbots plus puissants dans le cadre de ses travaux de recherche, Christophe Nicolle nous a demandé le semestre dernier d'effectuer un travail de recherche sur ce phénomène, et de développer notre propre chatbot. C'est ainsi qu'est né Checkbot.

Pour ce projet, nous n'avions pas à développer d'outil de traitement du langage, mais à en trouver un en effectuant un travail de recherche. Nous avons donc à l'époque décidé d'utiliser un outil appelé RASA. Une fois le cerveau de notre chatbot trouvé, il a fallu lui indiquer où trouver les connaissances, sa mémoire si on veut continuer l'analogie. Nous avons donc utilisé une base de connaissances appelée DBpedia. Il s'agit d'un projet universitaire ayant pour but de rassembler toutes les connaissances de Wikipédia dans un environnement facilement compréhensible par la machine. Par-dessus tout cela, nous avons développé les

liens entre toutes ses fonctions, ses nerfs en quelque sorte, et un système de « tchat » pour pouvoir communiquer avec lui.

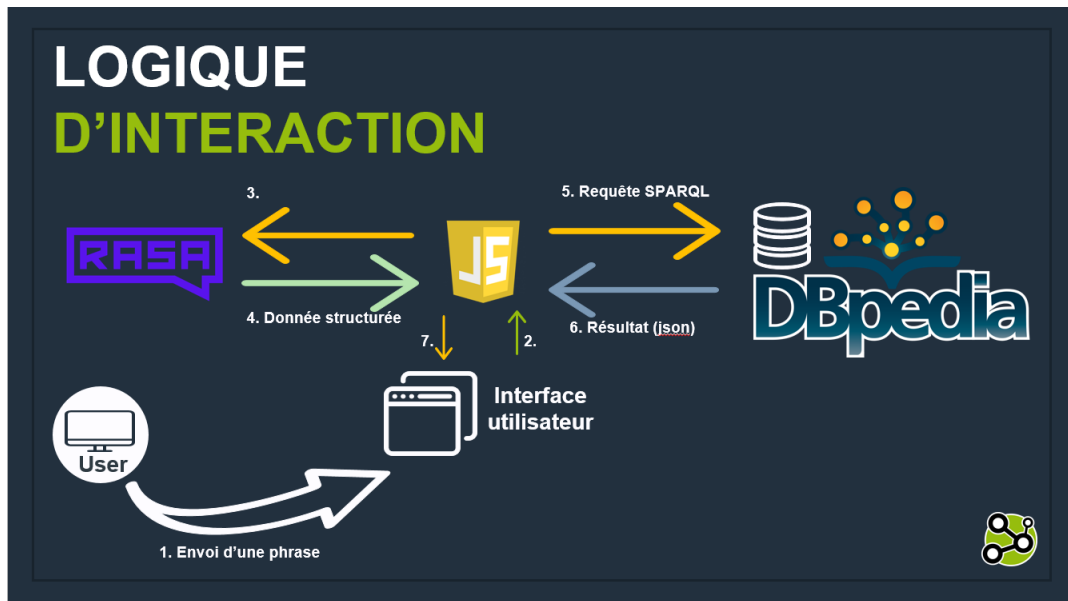


Figure 3 - Fonctionnement interne du chatbot - JS correspond à Javascript, le langage utilisé pour faire le lien entre les technologies

Checkbot se décline en deux versions, une version française et une version anglaise, et peut chercher la définition d'un mot, ses traductions dans de nombreuses langues, ainsi que ses synonymes pour la version anglaise. Il est également capable de mettre en forme toutes ces informations pour les rendre plus visuelles, et de saluer l'utilisateur quand celui-ci le fait.

III. PRÉSENTATION DES OBJECTIFS

Dans le but de creuser un peu plus loin ce sujet et afin de me préparer à mon stage dans lequel je vais devoir travailler à nouveau sur le développement d'un chatbot, j'ai décidé de continuer le développement de Checkbot, et ainsi de créer sa grande sœur, Oracle. Et afin de briser le suspense, non le nom ne vient pas de la société d'informatique notamment connue pour ses bases de données, mais d'un personnage de la firme DC Comics souvent réquisitionné par les autres pour obtenir des informations.

Alors que l'objectif du précédent semestre était de développer un outil, celui de ce semestre était de construire un assistant, un programme plus proche de l'être humain. Pour ce faire, j'ai donc décidé qu'il fallait que je travaille sur deux types d'améliorations : Etendre le champ de connaissances du chatbot et améliorer son système d'interaction avec l'être humain. Il fallait également que je lui trouve une nouvelle maison, un nouveau serveur, puisque celui utilisé précédemment allait arriver à expiration. Enfin, puisque le projet précédent a été fait avec six autres camarades, certains points de la conception du Checkbot m'étaient encore inconnus ou flous. Il fallait donc aussi se mettre à niveau sur tous ces points, pour avoir une vision complète et exacte du fonctionnement de notre chatbot.

Après un moment de réflexion, j'ai établi une première liste d'objectifs :

- Partie recherche
 - Se mettre à niveau sur le chatbot.
 - Lister les bases de connaissances ou les API² requêtables.

² API : Bibliothèque ou application mise à la disposition de tout le monde pour obtenir certaines informations ou effectuer certaines procédures.

- Rechercher une bibliothèque de synthèse et de reconnaissance vocale.
- Partie développement
 - Ajouter les nouvelles intentions possibles.
(= *Faire en sorte que le cerveau du chatbot reconnaisse plus d'intentions de l'utilisateur.*)
 - Implémenter les systèmes de requêtage.
(= *Créer les « nerfs » entre les nouvelles intentions du « cerveau » et la « mémoire » du chatbot.*)
 - Implémenter la synthèse et la reconnaissance vocale.
 - Mettre à jour l'interface du chatbot.
- Autre
 - Rechercher et acheter un nouveau serveur
 - Faire les installations nécessaires

Cette liste n'était pas exhaustive mais devait me servir de fil directeur au long du projet. Mon maître de stage avait également proposé de me rajouter quelques idées, afin de me préparer au mieux pour mon stage. Cependant, après avoir pris connaissance du sujet comme décrit ci-dessus, il a jugé que bien approfondir toutes ces améliorations serait plus bénéfique que d'en rajouter.

DEUXIÈME PARTIE : TRAVAIL EFFECTUÉ

I. ORGANISATION

Pour me guider à travers ce projet, j'ai utilisé plusieurs principes étudiés au cours de mon DUT concernant la gestion de projet. En effet, même si le fait d'être seul sur un projet permet certaines libertés concernant l'organisation, il était important de conserver certaines habitudes de gestion de projet afin de pouvoir juger de mon avancement et surtout afin de ne pas me perdre pendant la période de travail.

Pour ce faire, j'ai mis en place dès le début de mon projet une liste de tâches à effectuer. Cette liste est celle présentée [ici](#). Pour chaque, j'ai estimé une durée, puis je les ai réparties sur un planning, en prenant en compte le fait que je pouvais travailler jusqu'au 25 mars sur l'ancien serveur (celui utilisé pour le S3), et qu'il était plus intelligent de beaucoup charger les premières semaines plutôt que les dernières, puisque celles-ci allaient déjà être très chargées avec le reste des cours. Ce planning va de jeudi en jeudi parce qu'il s'agissait du jour de la semaine où je pouvais consacrer le plus de temps au projet tutoré, et n'est pas quotidien puisqu'il était assez peu probable que je puisse passer chaque soir un certain temps sur le chatbot (Environ 70% du travail de la semaine était fait le jeudi, 20% le week-end, et 10% le reste de la semaine). Vous pourrez retrouver ce planning en annexe ([annexe 2](#)).

II. ACHAT ET MISE EN PLACE D'UN SERVEUR

1. LE CHOIX DU SERVEUR

Comme énoncé précédemment, je me suis retrouvé dès le début de mon projet confronté à un problème : Le serveur sur lequel nous avons développé le Checkbot allait bientôt arriver à expiration, et il allait donc falloir en trouver un nouveau, adapté aux moyens d'une personne seule.

Concernant les critères de recherche, il était indispensable de trouver un serveur dit dédié, c'est-à-dire sur lequel on avait tous les droits. Ce point était important pour installer et faire fonctionner RASA, l'outil de traitement du langage du chatbot. De plus, il ne fallait pas qu'il soit excessivement cher, sans quoi mon portefeuille aurait été triste.

Au cours de mes recherches, deux hébergeurs se sont démarqués : OVH et 1&1. OVH avait l'avantage de ne pas être trop cher pour des serveurs simples, mais la plateforme ne proposait pas de serveurs dédiés à un prix raisonnable. C'est en grande partie pour cette raison que je me suis dirigé vers 1&1.

2. DE RASA À LUIS : CHANGEMENT DE CERVEAU

Une fois le serveur acheté, il a fallu commencer les différentes installations de Rasa pour faire fonctionner le chatbot. Et c'est à ce moment que les premières difficultés sont apparues. En effet, afin de fonctionner sur un serveur, l'outil avait besoin d'un proxy. Un proxy peut être vu comme une déviation sur la route. On dit aux requêtes qui entrent sur le serveur de prendre un autre chemin que celui qu'elles prennent en temps normal. Lors de la mise en place de ce dit proxy, je me suis retrouvé confronté à un problème avec le pare-feu de mon serveur. N'ayant pas de connaissances très poussées dans l'administration d'un serveur, j'ai contacté le service client de 1&1 et ensemble nous avons essayé de trouver une solution. Après quelques heures de recherche, nous n'avions toujours pas pu identifier d'où venait le problème, même si le technicien soupçonnait que le problème vienne directement de Rasa.

Je me suis alors ensuite tourné vers Dorian NAAJI, un des camarades avec qui j'avais travaillé le semestre précédent sur le Checkbot, et surtout celui qui c'était occupé de l'installation et de la mise en fonctionnement de Rasa. Ensemble, nous avons encore une fois cherché pendant quelques heures une solution, en vain. Suite à ces deux échecs pour trouver une solution, Dorian m'a conseillé de regarder Luis.ai, un outil de traitement du langage développé par Microsoft. Il avait à l'époque étudié cette technologie lors de notre phase de recherche mais nous avons finalement décidé de ne pas le garder parce que le traitement se faisait sur les serveurs de Microsoft, avec un nombre de requêtes au « cerveau » de Luis limité par jour (sans abonnement), en bref une solution qui n'était pas viable pour une utilisation professionnelle demandée à l'époque par M. Nicolle.



Figure 4 - LUIS, l'outil de traitement du langage naturel de Microsoft

Cependant, ceci n'était plus vraiment un problème aujourd'hui puisque le chatbot allait en grande partie être destiné à n'être utilisé plus que quelques fois à vocation pédagogique. J'ai donc décidé de me séparer de l'outil RASA pour utiliser LUIS. Il a ainsi fallu réécrire une partie du code pour pouvoir intégrer ce changement, mais tout fonctionnait correctement après cela.

III. ORACLE, PARLE-MOI ...

1. LA VOIX DU SAVOIR

Afin de briser toujours plus ce mur entre l'homme et la machine, l'idée de donner une voix à Oracle est venue assez naturellement. En effet, celle-ci pourrait ainsi formuler une réponse à l'oral en plus de la réponse écrite afin de créer un lien avec l'utilisateur. Dans cette optique, je me suis lancé à la recherche d'une bibliothèque Javascript³ me permettant à la fois d'implémenter un système de synthèse vocale (aussi appelée text-to-speech, tts) et de reconnaissance vocale (voir [partie suivante](#)). J'ai dressé le tableau suivant afin de comparer les avantages et inconvénients de chaque technologie trouvée.

Nom technologie	TTS Javascript	Artyom.js	ResponsiveVoice.js	meSpeak.js
Synthèse vocale				
Reconnaissance vocale				
Facilité d'utilisation				
Documentation bien fournie				

Après comparatif, Artyom.js s'avérait être une bibliothèque très intéressante pour mon projet, en grande partie parce qu'elle me permettait d'implémenter la synthèse et la reconnaissance vocale avec la même technologie, ce qui est un gain de place pour réduire la taille du chatbot, de facilité de lecture et d'implémentation dans le code puisqu'on évite de multiplier inutilement les objets à manipuler, et finalement

³ En informatique, une bibliothèque est un ensemble de fonctionnalités déjà implémentées. Javascript est un langage de développement, et est dans notre cas celui utilisé pour le chatbot.

un gain de temps pour moi puisqu'il n'y avait à apprendre à se servir plus que d'une bibliothèque. De plus, même si son utilisation allait rester dans le cadre d'un projet pédagogique, Artyom.js est soumise à une licence totalement libre et est donc open source (on peut la copier, la modifier et la distribuer comme on le souhaite). Grâce à sa facilité d'utilisation et à sa documentation complète et bien rédigée, l'implémentation de la synthèse vocale n'a pas posé de problème particulier. Un bouton afin de couper la parole à Oracle a été rajouté, pour ceux qui ne voudraient pas être dérangés par la synthèse.

Depuis ce semestre, Oracle lit maintenant tous les résultats qu'elle obtient et affiche à l'écran. Pour les traductions, le chatbot adapte sa voix à la langue retournée, afin d'avoir le meilleur accent possible. De plus, Oracle salue maintenant automatiquement l'utilisateur lorsqu'il se connecte à son interface.

2. LA RECONNAISSANCE VOCALE

Une fois le chatbot doté d'une bouche, il fallait maintenant lui rajouter des oreilles, afin de pouvoir entendre les questions que nous avons à lui poser. Toujours avec Artyom.js donc, l'implémentation n'était pas bien plus compliquée que pour l'amélioration précédente. Cependant, un autre problème a fini par faire surface : La voix est considérée par la loi comme une donnée personnelle. Cela pose dans notre cas un faux et un vrai problème. En effet, le faux problème concerne l'acquisition de cette donnée. Pour résoudre ce problème, il suffisait de demander l'avis de l'utilisateur avant de commencer la prise de son. Il s'agit d'un faux problème puisque tous les navigateurs actuels demandent déjà automatiquement l'avis de l'utilisateur avant d'acquérir une donnée personnelle. Le vrai problème concerne l'infrastructure autour de la réception de celle-ci. En effet, la loi exige qu'une donnée sensible soit manipulée par une structure sécurisée. Pour un site web, cela consiste à posséder un certificat SSL, une protection délivrée par des organismes officiels. Une fois ce certificat obtenu, le site n'est plus référencé en HTTP mais en HTTPS. Or, obtenir un tel certificat prend plusieurs mois et coûte une somme non-négligeable d'argent. Passer en HTTPS n'était donc pas une solution viable.

Il allait donc être impossible d'implémenter sur mon serveur la reconnaissance vocale. Cependant, il aurait été dommage d'avoir dédié autant de temps à un problème sans issue, c'est pourquoi j'ai pris à ce moment-là la décision de développer en parallèle deux versions d'Oracle : Une version fonctionnant en http, et une version adaptée pour le HTTPS. En effet, tous les problèmes causés par l'utilisation de données personnelles ne sont plus des problèmes quand on travaille en localhost, c'est-à-dire dans un environnement où notre ordinateur joue le rôle d'un serveur web. Dans ce cas, tout fonctionne donc comme si on travaillait en HTTPS, mais le site est évidemment factice et n'est pas disponible pour une autre personne que celle qui a accès à l'ordinateur.

Pour améliorer le rendu de la reconnaissance, j'ai également ajouté une petite forme qui réagit au son de la voix en arrière-plan pendant l'enregistrement du son. Cela permet de visualiser les moments où Oracle reconnaît le son de la voix.

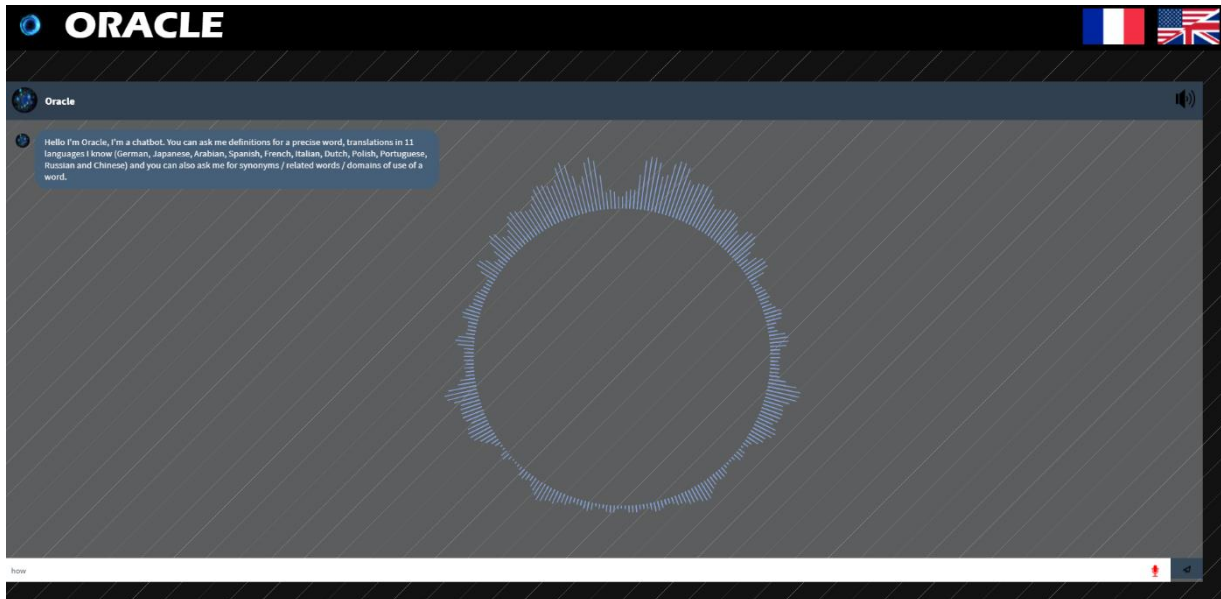


Figure 5 - Quand Oracle écoute, le micro dans la barre de recherche devient rouge, une forme se crée et s'anime et le texte s'écrit tout seul dans la barre

IV. ORACLE, QUEL TEMPS FAIT-IL ?

1. RÉCUPÉRER LA MÉTÉO

Notre chatbot possédant maintenant un cerveau, une bouche, des oreilles et des connexions nerveuses pour relier le tout, il était plus que temps d'élargir son champ de connaissances. Pour cela, et toujours dans l'objectif de faire d'Oracle un assistant, j'ai décidé de lui apprendre à se renseigner sur la météo. J'ai pour cela utilisé l'outil OpenWeatherAPI, une application requêteable depuis un site web par exemple, et qui peut retourner la météo actuelle et les prévisions jusqu'à 14 jours d'un endroit précis. Il s'agit de l'outil le plus connu dans le domaine, qui possède les mêmes avantages et inconvénients de [Luis.ai](https://luis.ai), c'est-à-dire ne nécessiter aucune installation mais en même temps n'être requêteable qu'un nombre limité de fois par jour. Il faut également savoir que cette API⁴ remonte les prévisions par tranche de deux heures, c'est-à-dire que si on lui demande les prévisions météorologiques pour le lendemain, elle retournera douze résultats, et que cela comptera comme douze requêtes. En fonction du nombre d'utilisateurs amenés à utiliser le projet à la fin, il fallait donc ne pas demander une prévision trop lointaine.

Pour commencer, j'ai choisi de faire des prévisions à l'instant : Oracle se renseigne sur le temps qu'il fait dehors, récupère certaines informations utiles et les met en page. Au final, j'ai décidé de ne pas implémenter de prévisions plus lointaines, essentiellement à cause d'un manque de temps. Cependant, la version française comme la version anglaise du chatbot sont capables de récupérer la météo, et Oracle fait une bien meilleure présentatrice météo que Catherine Laborde. On peut aussi noter que la couleur de la température et l'image de présentation de la météo varient en fonction du temps qu'il fait dehors :

- < 0°C : Bleu clair
- Entre 0 et 10°C : Bleu

⁴ API : Bibliothèque ou application mise à la disposition de tout le monde pour obtenir certaines informations ou effectuer certaines procédures.

- Entre 11 et 20°C : Or
- Entre 21 et 30°C : Orange
- > 31°C : Rouge



Figure 6 - Rendu de la fiche météo éditée par Oracle

2. LOCALISER L'UTILISATEUR

Les principales difficultés concernant cette amélioration sont venues au moment de localiser l'utilisateur. Au départ, deux scénarios étaient conçus : L'utilisateur demande la météo et précise la ville dans laquelle il se trouve, ou alors l'utilisateur demande juste la météo. Dans le deuxième cas, pas d'ambiguïté, il suffisait de demander à OpenWeatherAPI le temps qu'il faisait dans la ville demandée. Pour pouvoir faire cela, l'outil demandait les coordonnées spatiales de la ville (latitude, longitude). Après quelques recherches sur la manière de récupérer les coordonnées d'un lieu, j'ai fait le choix de questionner l'API de Google Maps, qui possède l'avantage de fonctionner de manière assez semblable à OpenWeatherAPI, et surtout qui possède une énorme base de données recensant la plupart des villes du monde. Une fois cela fait, j'avais en ma possession toutes les informations dont j'avais besoin : je demandais à Google les coordonnées de la ville demandée par l'utilisateur et je récupérais la météo de cet endroit via OpenWeatherAPI. Le scénario problématique était plutôt le deuxième, et pour une raison qui était loin de m'être étrangère. En effet, lorsque l'utilisateur demandait uniquement la météo au chatbot, j'avais prévu que celui-ci soit capable de géolocaliser son appareil, et de demander directement à l'API la météo à l'endroit obtenu. Et c'est là que l'on se retrouve dans une situation assez familière : la position d'une personne est reconnue comme une donnée privée et donc sensible, soit le même problème que pour [la reconnaissance vocale](#). Bien conscient du problème au moment d'implémenter cette fonctionnalité, j'ai donc décidé de revoir légèrement mon système. Si le chatbot se retrouve dans ce scénario-ci, il demande à l'utilisateur de préciser dans quelle ville celui-ci se trouve. Et lorsque l'utilisateur entre cette précision, on retombe sur le scénario précédent. De fait, j'ai tout de même

appris comment demander une géolocalisation à une personne via Javascript, même si cette fonctionnalité n'a jamais été implémentée directement sur le chatbot, même sur la version HTTPS.

V. ORACLE, SOURCE DE SAVOIR

1. QUI EST CETTE PERSONNE ?

Enfin, toujours dans le but d'élargir les connaissances du chatbot, j'ai décidé de lui apprendre à chercher des informations concernant une personne dans la base de connaissances DBpedia, qui est une version optimisée pour les machines de Wikipédia (il faut comprendre par là qu'il s'agit d'une énorme base de connaissances). Une fois ces informations récupérées, Oracle les arrange sous forme de petites fiches synthèses, comme on peut en voir sur Wikipédia ou plus récemment sur Google.



Figure 8 – Petites fiches synthèses proposées par Google

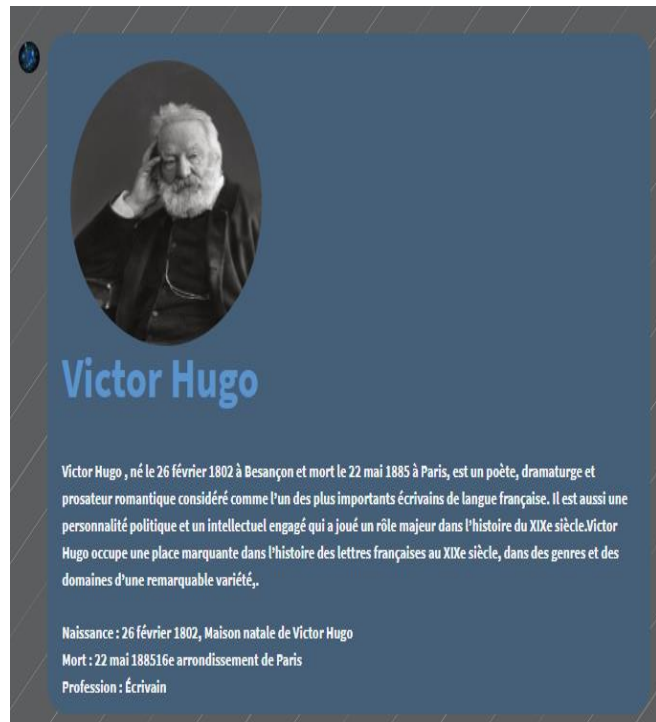


Figure 7 - Exemple de fiche synthèse proposée par Oracle

Contrairement aux requêtes à DBpedia implémentées le semestre dernier pour obtenir des informations sur un mot, celle utilisée dans ce cas est beaucoup plus modulable et s'adapte totalement à chaque type de personne. En effet, il est important de comprendre que chaque personne sur DBpedia n'est pas définie de la même manière, et possède donc des informations que d'autres n'auront pas. Par exemple, Nelson Mandela possèdera une date et un lieu de mort, contrairement à Tom Cruise qui est toujours en vie.

De même, certains seront mariés à une personne, alors que d'autres non, etc. La requête SPARQL⁵ répondant à ce besoin prend en compte ces spécifications, essaiera à chaque fois de remonter un maximum d'informations, mais n'échouera pas pour autant si elle n'arrive pas à tout récupérer.

La difficulté associée à cette amélioration a été justement de rendre de la même manière que la requête le code du chatbot totalement adaptable en fonction du nombre d'informations remontées. Pour résoudre ce problème, il a suffi de penser à vérifier avant l'affichage d'une information que celle-ci existe. Au maximum, le chatbot est capable de récupérer toutes les informations suivantes :

- Nom
- Date et lieu de naissance
- Date et lieu de mort
- Courte biographie
- Photographie
- Métier
- Occupation principale
- Nationalité
- Epoux(se)

2. CONNAIS-TU CETTE VILLE ?

Dans la même veine que ce qui a été fait précédemment, Oracle est capable de rédiger une fiche synthèse sur les informations qu'elle a pu obtenir sur une ville. Toutes les remarques faites précédemment sont aussi valables pour cette partie, puisque les deux ont été implémentées en même temps et presque de la même manière (Demander des informations sur une ville se fait de la même manière que sur une personne, mais les informations remontées sont loin d'être les mêmes, elles). Au maximum, le chatbot est capable de récupérer toutes les informations suivantes :

- Nom
- Code Postal
- Département
- Région
- Pays
- Nombre d'habitants
- Altitude (min et max)
- Courte description
- Coordonnées (latitude, longitude)
- Nom du maire
- Nom des habitants

⁵ Langage de requêtage utilisé pour récupérer des informations sur DBpedia

- Photographie



Figure 9 - Fiche synthèse sur une ville

VI. AMÉLIORATIONS MINEURES

1. MODIFICATION DE L'INTERFACE

Afin de démarquer cette version du chatbot avec la précédente, il était important de revoir la charte graphique et plus généralement l'aspect de l'interface d'Oracle. Alors que les couleurs principales du Checkbot étaient le vert, le gris et le noir, j'ai choisi que celles d'Oracle seraient le bleu, le noir et le blanc.

La principale difficulté de ces changements était de s'imprégner dans le code de quelqu'un d'autre, puisque l'interface du site (le fichier HTML et les CSS pour la mise en page) avait été codée par un autre membre de mon groupe. Cela m'a permis d'approfondir mes connaissances en CSS notamment, type de fichier assez peu travaillé en DUT et qui permet pourtant de totalement faire la différence entre deux sites au contenu identique.

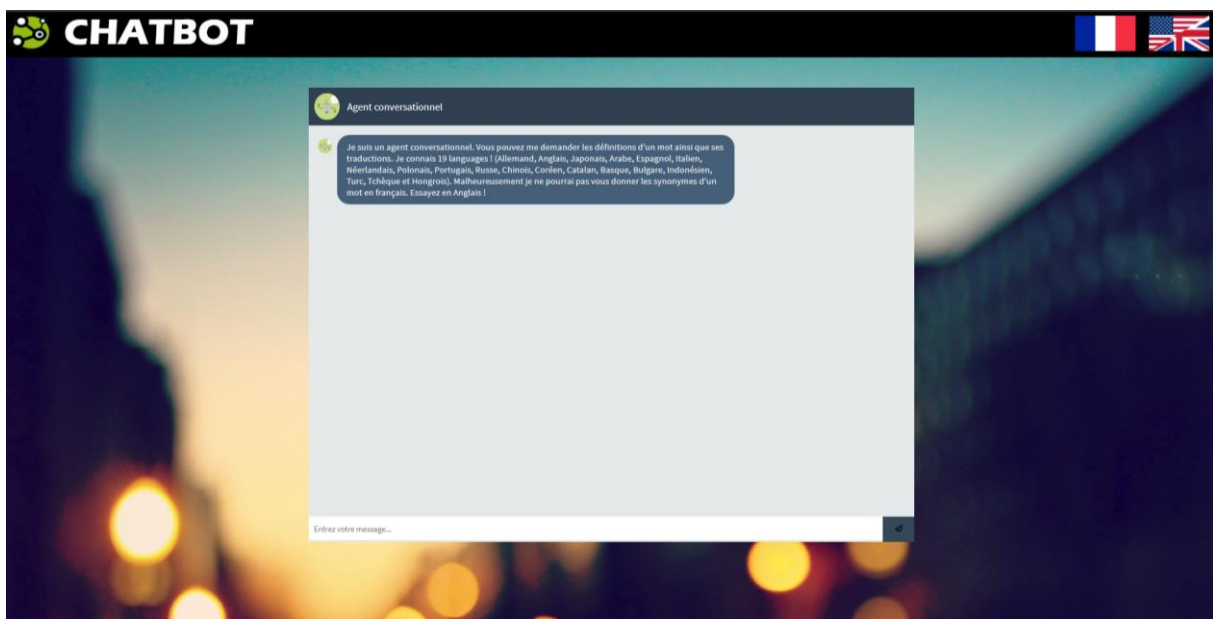


Figure 10 - Interface du checkbot (avant)

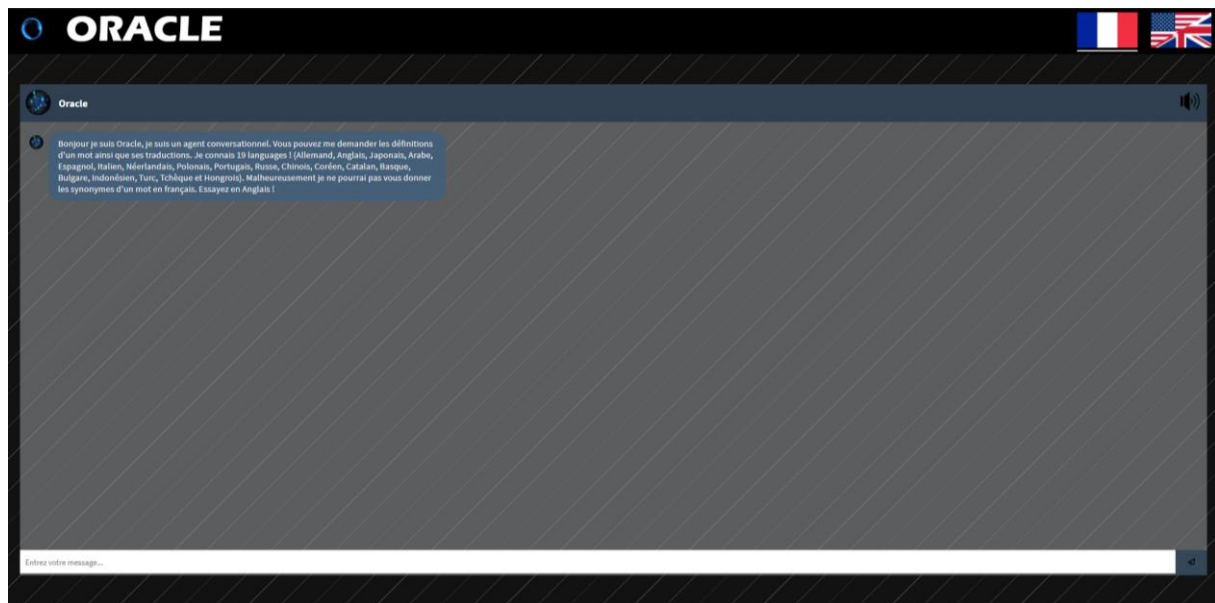


Figure 11 - Interface d'Oracle (après)

2. MISE EN PLACE D'UN HISTORIQUE

Lors du développement des précédentes extensions, j'ai pu remarquer qu'il pouvait parfois être gênant de devoir toujours taper le même genre de questions au chatbot. J'ai alors mis en place un système d'historique, qui garde en mémoire les 20 derniers messages envoyés par l'utilisateur au chatbot. Cet historique est valable pour une session entière sur le site, c'est-à-dire tant que la page n'est pas fermée ou rechargée. On remonte ou descend dans la liste des messages en appuyant sur les flèches « Haut » et « Bas » du clavier. Cette amélioration n'a pas posé de problème particulier.


TROISIÈME PARTIE : POUR CONCLURE ...
















I. COMPETENCY QUESTIONS - RÉSULTATS

Dans le monde de l'intelligence artificielle, on a tendance à représenter les connaissances d'un programme par une liste de questions concernant ce qu'il peut faire. Ces questions sont à chaque fois évaluées par un code couleur qui indique si le programme sait faire quelque chose, s'il ne sait pas le faire, ou s'il peut encore faire des progrès (de la même manière qu'on évalue les enfants en école primaire avec des « Bien », « A revoir » et « Non-acquis »). Voici la liste des competency questions d'Oracle.

 Acquis

 A revoir

 Non-acquis

Le chatbot peut-il saluer l'utilisateur ?	
Le chatbot peut-il donner la définition d'un mot sans mise en forme ?	
Le chatbot peut-il remonter les traductions d'un mot sans mise en forme ?	
Le chatbot peut-il remonter les synonymes d'un mot sans mise en forme ?	
Le chatbot peut-il donner les mots apparentés du mot sans mise en forme ?	
Le chatbot peut-il répondre aux tâches précédentes avec une mise en forme quelconque ?	
Le chatbot est-il capable de mettre en forme les traductions d'un mot sous forme d'un tableau à 2 colonnes : - Colonne 1 : Mot traduit - Colonne 2 : Langue + éventuellement un drapeau	
Le chatbot est-il capable de mettre en forme les synonymes et mots apparentés d'un mot sous forme d'une carte mentale ?	
Le chatbot est-il capable de restituer les informations concernant un mot en comprenant le type d'information que veut l'utilisateur ? (Définition si l'utilisateur demande une définition, synonyme(s) si l'utilisateur demande des synonymes, etc.)	
Le chatbot est-il capable de formuler une réponse à l'oral ?	
Le chatbot adapte-t-il son accent en fonction de la langue qu'il lit ?	
Le chatbot est-il capable d'écouter et de noter la question de l'utilisateur ?	
Le chatbot peut-il se taire quand on le lui demande ?	
Le chatbot est-il capable de récupérer la météo dans un lieu précisé ?	
Le chatbot est-il capable de récupérer la météo à l'endroit où se trouve l'utilisateur ?	

Le chatbot est-il capable de mettre en forme les informations concernant la météo ?	
Le chatbot est-il capable de récupérer des prévisions météorologiques des jours à venir ?	
Le chatbot est-il capable de récupérer des prévisions météorologiques des heures à venir ?	
Le chatbot est-il capable d'enregistrer les dernières requêtes de l'utilisateur ?	
Le chatbot propose-t-il un système d'historique ?	
Le chatbot peut-il remonter quelques informations sur une personne ?	
Le chatbot peut-il remonter quelques informations sur une ville ?	
Le chatbot peut-il mettre en forme un ensemble d'informations sous forme de synthèses ?	

II. BILAN DES COMPÉTENCES ACQUISES

Puisque les bases de ce projet étaient déjà existantes, il a été possible de très vite se concentrer sur la conception et le développement de nouveautés et d'améliorations jamais travaillées auparavant. De fait, de nombreuses compétences techniques ont été acquises en très peu de temps : Manipuler les fonctionnalités audios d'un programme (synthèse et reconnaissance), géolocaliser une personne, se servir d'une API⁶ en ligne pour obtenir des informations, créer des animations en Javascript, ... Approfondir ce sujet m'a également permis d'en prendre pleinement conscience, puisque j'ai dû pour le mener à bien apprendre et comprendre en très peu de temps tout ce que mon équipe du dernier semestre avait appris et fait, dans les moindres détails. Je pense par exemple à tout ce qui entoure le traitement du langage naturel (Rasa, Luis, ...), technologies que j'avais étudiées en surface pendant le dernier semestre, mais dont certaines subtilités m'étaient encore floues voire inconnues (définition des intentions, désambiguïsation, ...).

Continuer le développement de ce chatbot a également été un bon complément aux matières de **Client Web** (manipulation de Javascript, langage dans lequel est codé Oracle) et de **Web Sémantique** ([requêtage SPARQL](#), [étude de DBpedia](#)) étudiées lors du quatrième semestre. En effet, parfois les problématiques vues en cours m'ont permis d'avancer sur le projet, et parfois les connaissances acquises pendant mes recherches personnelles pour le chatbot m'ont permis de mieux comprendre, voire de me débloquer sur certains exercices. Ainsi, travailler sur Oracle a contribué à approfondir mes connaissances sur les langages du web (HTML pour la structure de la page, CSS pour l'esthétique et Javascript pour l'interactivité avec l'utilisateur) mais aussi sur certains principes fondamentaux du web sémantique comme le SPARQL, les bases de connaissances, les ontologies, le vocabulaire contrôlé, etc.

Enfin, les différentes difficultés rencontrées sur ce projet ont été plus que constructives et utiles. En effet, devoir commander mon propre serveur web et le configurer pour régler les problèmes de pare-feu ([partie 2, II](#)) m'a permis d'améliorer et/ou d'acquérir certaines compétences en administration de serveur, notamment pour tout ce qui concerne l'utilisation de proxys, de règles de pare-feu et l'installation de technologies sur un serveur. Quant à mes problèmes concernant les données sensibles, cela m'a permis d'étudier un cas concret dans lequel l'informatique ne peut pas tout faire, et donc me sensibiliser sur le sujet.

⁶ API : Bibliothèque ou application mise à la disposition de tout le monde pour obtenir certaines informations ou effectuer certaines procédures.

J'ai pu également découvrir comment un serveur pouvait passer d'un protocole HTTP à un protocole HTTPS, et ce qu'est exactement un certificat SSL.

III. PROLONGATIONS POSSIBLES

De nombreuses prolongations pour ce projet sont possibles (puisque de fait les extensions possibles sont presque infinies : compréhension de nouvelles intentions, accès à de nouvelles données, ...). Cependant, quelques fonctionnalités mériteraient d'être explorées ou approfondies avant de se lancer dans l'implémentation de nouvelles.

Tout d'abord, je pense que le code mériterait d'être refactorisé correctement. En informatique, refactoriser un code correspond à le réécrire différemment, souvent pour l'améliorer ou pour changer de langage. Dans notre cas, le chatbot pourrait être réécrit facilement en Typescript, un langage dérivé de Javascript mais permettant de coder de manière plus « propre ». De même, une grosse partie du code ayant été codée par des étudiants de troisième semestre non-initiés au Javascript, certaines fonctions pourraient être simplifiées. Enfin et surtout, la conception et la structure même du projet devraient être revues afin de disperser un peu plus le code et éviter l'objet divin⁷ qui sert de script principal au projet, sans quoi M. Guidet pourrait venir à s'arracher les cheveux à la vue du code.

Il serait également intéressant de creuser la piste du deep learning, un principe d'intelligence artificielle où l'intelligence du programme s'améliore avec l'expérience et après chaque conversation avec un utilisateur, en retenant des habitudes d'utilisation, en faisant des liens entre ses fonctionnalités, etc. En traitement du langage naturel, il existe un concept appelé les « stories » qui correspond à des enchaînements d'actions, ce qui permet d'accentuer le côté « discussion » (Le chatbot est capable de communiquer avec l'utilisateur même si celui-ci ne dit rien). De même, on pourrait imaginer faire des liens entre les différentes fonctionnalités. Par exemple, depuis la fiche synthèse d'une personne, on voit que celle-ci est née dans telle ville, donc offrir la possibilité de cliquer sur le lieu pour afficher des informations avec la fonctionnalité sur les villes, et vice versa avec les maires. L'homogénéisation des compétences du chatbot et l'amélioration de l'intelligence artificielle seraient deux moyens de se rapprocher encore plus de l'assistant.

Ensuite, une des extensions travaillées ce semestre pourrait être exploitée encore plus : celle de la météo. On pourrait certes penser à passer en HTTPS, ce qui permettrait également d'implémenter la reconnaissance vocale sur le serveur, mais on pourrait surtout travailler le système de prévisions, de manière à ce qu'Oracle puisse prédire la météo sur une plus ou moins longue période (Par exemple demander le temps qu'il fera à tel endroit tel jour à tel moment de la journée).

Enfin, il serait également intéressant de mener une réflexion et un travail complet sur la désambiguïsation des intentions. En effet, plus on multiplie le nombre d'intentions compréhensibles par le chatbot, plus le risque qu'il les confonde devient grand. Typiquement, certaines formulations telles que « Que peux-tu me dire à propos de ... » peuvent être suivies par un mot, par un nom de ville ou par un nom de personne. Dans ce genre de cas, le chatbot peut se retrouver perdu au moment de déterminer de quelle intention il s'agit. Pour cela, on pourrait par exemple ajouter un nouveau mode au chatbot : quand celui-ci est indécis. Il pourrait alors poser certaines questions à l'utilisateur pour identifier le vrai sens de sa requête.

⁷ Mauvaise pratique de codage consistant à ne posséder qu'un seul fichier de code qui s'occupe de toutes ou de presque toutes les fonctionnalités du programme.

ANNEXES

I. TESTER ET UTILISER ORACLE

Pour tester la version principale d'Oracle (celle dont les fonctionnalités sont décrites [ici](#)), rien de plus simple. Vous pouvez en effet accéder à l'interface du chatbot en cliquant sur ce [lien](#). Vous arrivez ainsi sur la version française du programme. Pour passer à la version anglaise, il suffit de cliquer sur le drapeau mi-Royaume-Uni mi- Etats-Unis. Vous pouvez écrire une question dans la barre prévue à cet effet, et confirmer en appuyant sur « Entrée » ou en cliquant sur la flèche à côté. Vous pouvez désactiver la voix d'Oracle en cliquant sur le haut-parleur noir en haut à droite de la fenêtre de « tchat ».

J'ai pris la décision de ne pas mettre à disposition la version « localhost » du chatbot (celle implémentant la reconnaissance vocale) car je n'ai pas trouvé de moyen simple de la faire installer sur un autre ordinateur (sans installer de logiciel tiers notamment). Cependant, vous pouvez trouver une démonstration [ici](#).

Enfin, même si vous êtes libres de faire les tests de votre choix avec Oracle, voici quelques questions types permettant à coup sûr d'obtenir des résultats convaincants :

- Sur la VF :
 - Bonjour Oracle (salutations)
 - Comment dit-on arbre en anglais, en espagnol et en russe ? (traductions)
 - Qu'est-ce qu'une chaise ? (définition)
 - Quel temps fait-il à Dijon ? (météo)
 - Est-ce qu'il fait froid ? puis entrez le nom d'une ville (météo)
 - Qui est Nelson Mandela ? (personne)
 - Que peux-tu me dire sur la ville Dijon ? (ville)
- Sur la VA :
 - Hello Oracle (greetings)
 - How can I say cat in french and italian ? (traductions)
 - What is a bag ? (definition)
 - How is the weather ? puis le nom d'une ville (weather)
 - Is it cold in Moscow ? (weather)
 - Who is Barack Obama ? (person)
 - Do you know the city Miami ? (city)

II. PLANNING D'ORGANISATION

Retour vers la partie « [Organisation](#) ».

Planning projet tutoré S4								
Semaine 1			Semaine 2			Semaine 3		
Du 02/02 au 07/02			Du 08/02 au 14/02			Du 15/02 au 21/02		
			Début vacances (très disponible)			Vacances (moyennement disponible)		
Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le
Mise en place de l'organisation			Tester et apprendre à se servir d'Artyom.js			Tester la reconnaissance vocale		
Réflexion sur les nouvelles intentions à implémenter			Implémenter la synthèse vocale			Implémenter la reconnaissance vocale		Semaine 4
Recherche d'une bibliothèque pour reco/synthèse vocale			Rechercher une API pour obtenir la météo					
			Apprendre à se servir de cette API		Semaine 3			
			Modifier l'IHM (apparence de l'interface)					

Semaine 4			Semaine 5			Semaine 6		
Du 22/02 au 28/02			Du 01/03 au 07/03			Du 08/03 au 14/03		
Vacances (peu disponible)								
Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le
			Acheter le nouveau serveur			Créer un projet Luis.ai		
			Effectuer toutes les installations nécessaires			Implémenter les intentions et les entités		
			Régler le problème avec Rasa		Semaine 6	Modifier le code du chatbot en conséquence		

Semaine 7			Semaine 8			Semaine 9		
Du 15/03 au 21/03			Du 22/03 au 28/03			Du 29/03 au 03/04		
			Examens			Déménagement		
Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le	Tâche	Terminée dans les temps	Sinon, terminée le
Implémenter la demande de météo			Fiche synthèse personne			Rapport		
Mettre en forme les résultats			Fiche synthèse ville					
Mise en place de l'historique			Correction de bugs		Semaine 9			