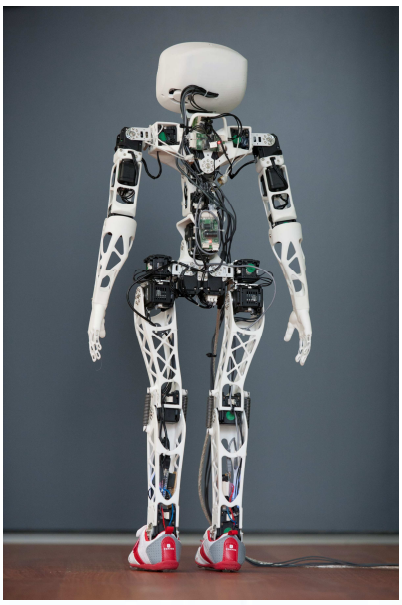
27/04/2017

Projet informatique individuel

Projet Cherry : Amélioration de la communication verbale du robot – ENSC 2A



Grégoire MELIN

BORDEAUX INP- ECOLE NATIONALE SUPERIEURE DE COGNITIQUE

TABLE DES MATIERES

INTRODUCTION

L'ASSOCIATION PRIMA

PRIMA est une association de bénévoles créée en 2004, régie par la loi 1901. Toute l'action de PRIMA s'articule autour de l'enfant ou du jeune adulte malade, pour rompre leur isolement. La mission que se fixe l'association est d'aider les enfants hospitalisés, les handicapés et les adolescents en difficulté à garder le contact par le biais d'internet grâce à des équipements, internet, et à des bénévoles. En d'autres termes, le but est de rompre l'isolement des enfants par la mise à disposition de technologie allant d'ordinateur à des robots capables d'interactions.

POSITIONNEMENT & OBJECTIFS

Dans les hôpitaux, certains enfants pris en charge peuvent rester dans leurs chambres plusieurs jours voir plus. Or pour un enfant, le temps est long et passer du temps dans un hôpital peut être inquiétant voir même traumatisant d'où l'idée de l'association PRIMA qui a créé le projet Cherry. Le projet Cherry est un projet communautaire visant à développer les usages en milieu hospitalier permettant de rompre l'isolement des enfants (maternelle jusqu'au collège) à travers la mise à disposition d'un robot humanoïde afin de les divertir et de leur tenir compagnie. Le projet utilise le robot POPPY, robot open-source et également communautaire afin de discuter avec l'enfant, jouer à des jeux ou encore assister le personnel hospitalier dans l'éducation thérapeutique. Le projet est divisé en chantiers qui constituent les différentes fonctionnalités du robot Cherry.

Fig. : Chambre pédiatrique standard de l'hôpital Villeneuve-Saint-George

EQUIPE PROJET & PREREQUIS TECHNIQUES

EQUIPE PROJET

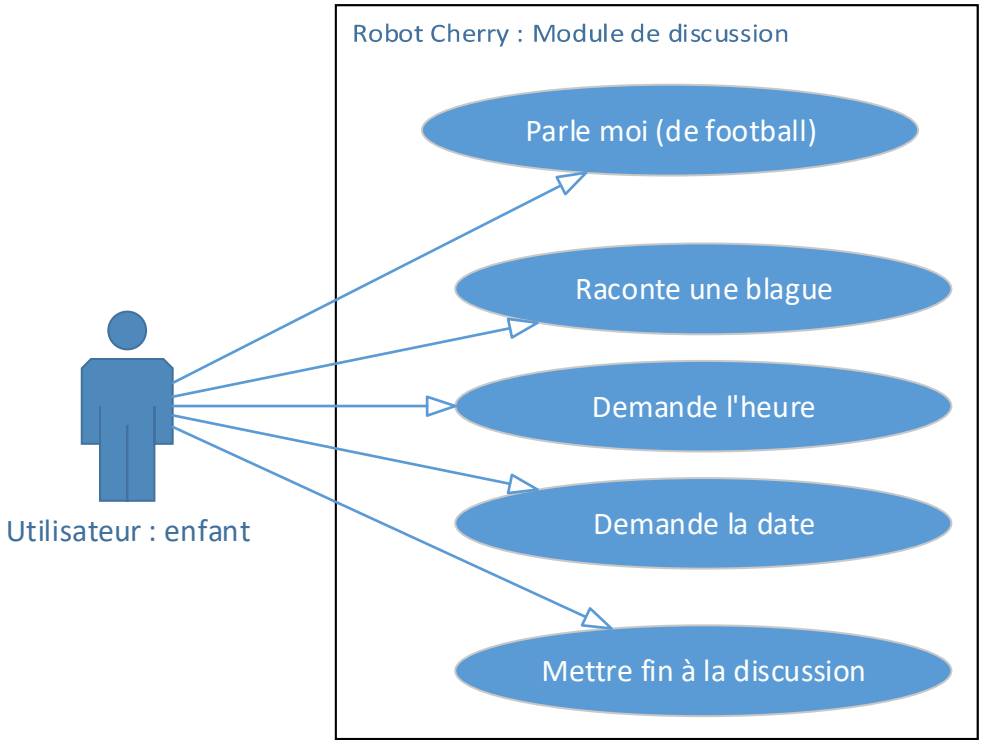
Le projet sera mené par Grégoire MELIN, élève de deuxième année à Bordeaux INP – Ecole Nationale Supérieure de Cognitique.

MATERIEL

D'un point de vue matériel, le robot Cherry, robot entièrement imprimé en 3D par des chercheurs de l'INRIA (Institut Nationale de Recherches en Informatique et Automatique) est animé par un microcontrôleur de type Odroid sur lequel est installé une version Debian de Linux à savoir un Ubuntu 16.04 LTS. Le choix a donc été fait de développer l'ensemble du projet en utilisant un OS du même type. L'ensemble du projet a été développé sur une machine virtuelle, les temps & les performances sont donc en fonction de ces prérequis. De plus, les API utilisant des ressources venant d'internet, les temps d'attente notamment lors de la reconnaissance vocale sont à mettre en corrélation avec la qualité de la connexion.

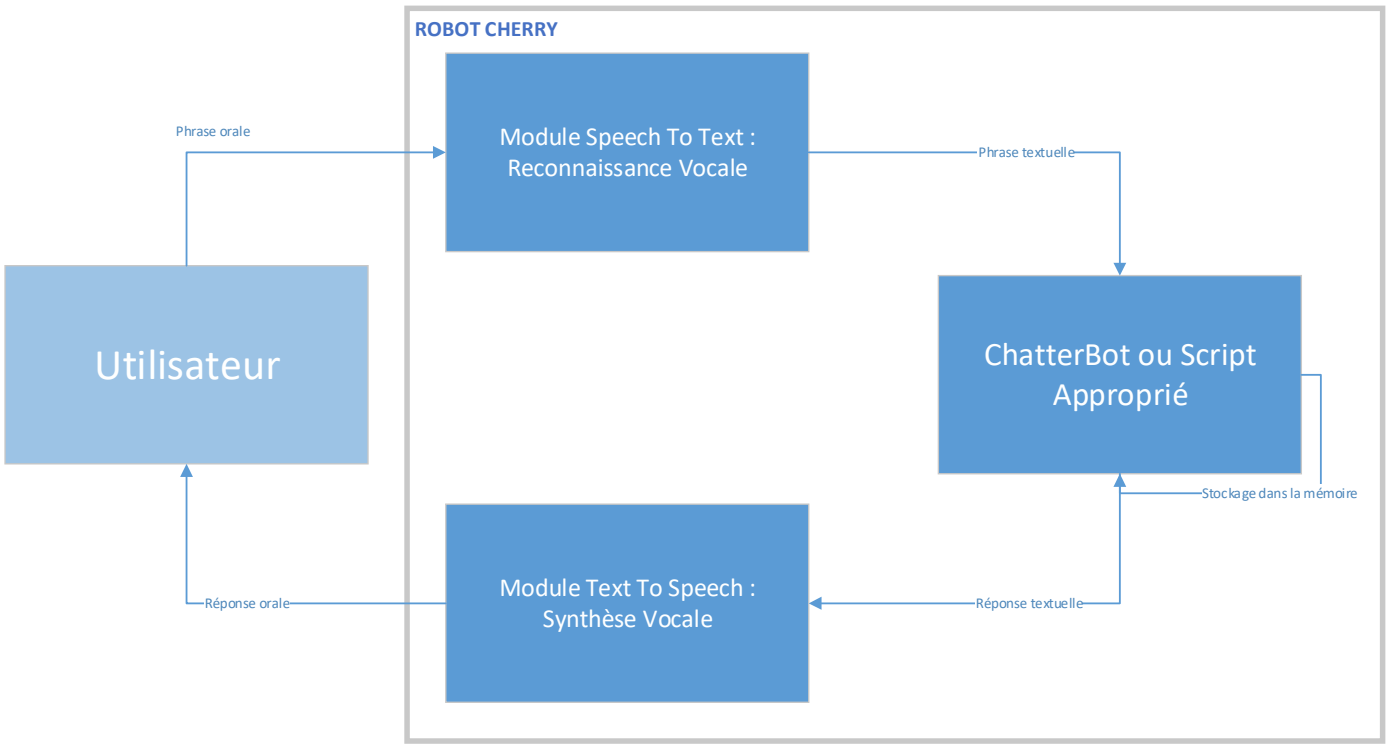
MODELISATION DU PROJET

CAS D'UTILISATION



STRUCTURE FONCTIONNELLE DE LA SOLUTION

La communication du robot Cherry « idéale » aux vues des choix de conception être modélisé de la sorte :



L'idée est donc de proposer une solution permettant une communication la plus « naturelle » possible.

FOCALISATION SUR DES OBJECTIFS PRECIS : SCENARIO D'UTILISATION

MISE EN PLACE D'UN PERSONNA

Soit un enfant faisant parti de la cible du projet :

Nom : Dupond

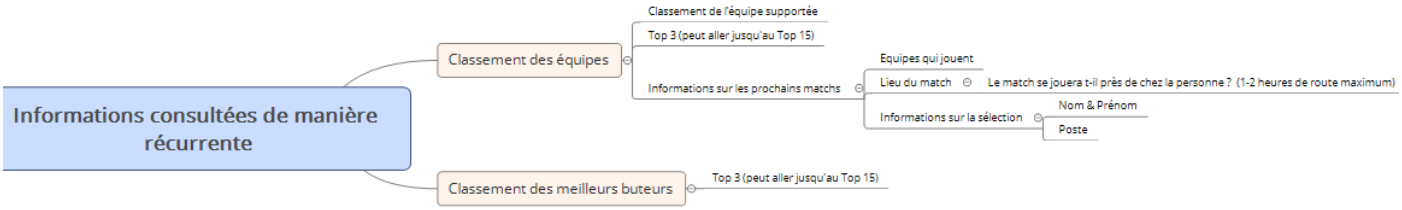
Prénom : Pierre

Age : 15

Centre d'intérêt : Foot

RESULTAT INTERVIEW

Il a été choisi de donner la possibilité au robot de parler de football et plus particulièrement de ligue 1. Afin de pouvoir sélectionner les informations que le robot peut donner de manière réaliste, le projet s'est focalisé sur des informations essentielles. Des interviews de fan de foot ont alors été faite dont le résultat est le suivant :



SCENARIO

Le projet étant un travail énorme irréalisable dans les délais, il a été convenu qu'on se focaliserait sur un cas d'utilisation. Soit un enfant Pierre Dupond, sortant du bloc suite à une lourde opération. Il se réveille donc. On lui apporte alors le robot Cherry sur un plateau. Le robot est démarré. Il se demande quel peut bien être cette chose qu'on lui a apporté il demande donc au robot qui il est. Pierre, rassuré, voudrait connaître en premier lieu la date et l'heure qu'il est puisqu'il a dormi pendant a priori plusieurs jours et les affichages de sa chambre et sa condition physique ne lui permettent pas d'avoir accès à ses informations. Suite à cela, il demande à Cherry de lui raconter une blague. Ne sachant plus trop quoi dire au robot il demande à ce dernier de lui parler de quelque chose. Le robot lui parle alors de football lui donnant les deux derniers matchs et les scores associés. Pierre est fatigué et décide donc de dormir. Il dit donc à Cherry de mettre fin à la discussion : Cherry s'éteint alors. Ce scénario est illustré dans la vidéo jointe au dossier.

REALISATION

ETAT DE L'ART

CHATTERBOT

L'API gérant le ChatBot est la bibliothèque *chatterBo*t, assez performante est entraînable via plusieurs mécanismes exploitables et intéressant. L'essentiel du projet consiste en l'implémentation d'interactions passant par une réaction prédéfinie à des stimuli prédéfinis. A ce jour, plusieurs pistes sont explorées.

MODULE TTS

En termes de synthèse vocale, une API candidate viable est *eSpeak*, bibliothèque open-source, disponible en français et donc tout à fait compatible avec les exigences du projet. Cependant, l'API Google est également performante. Ce qui différencie les deux est que l'API Google ne fait pas de streaming : La synthèse vocale nécessite la création d'un fichier audio (.mp3) et sa lecture.

MODULE STT

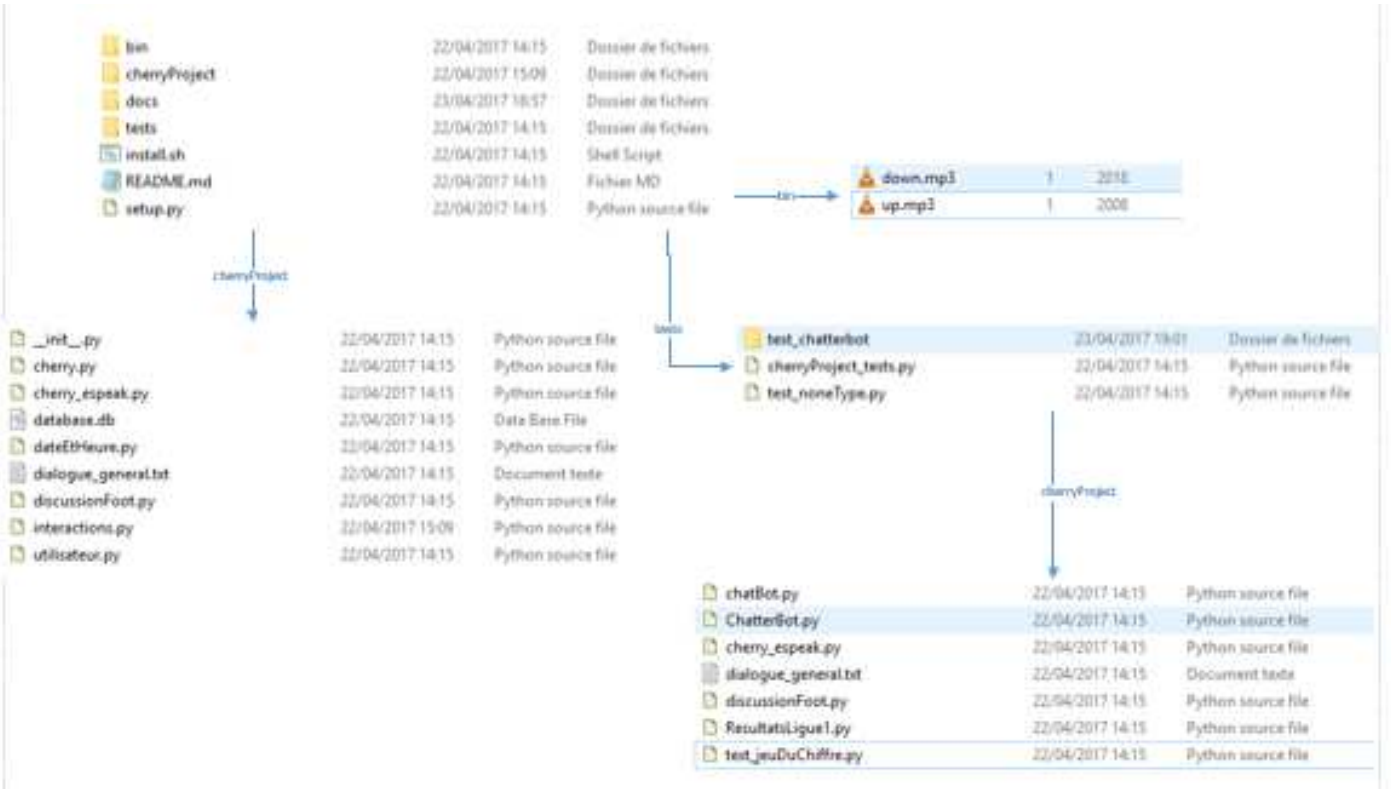
En termes de reconnaissance vocale, le problème est de trouver une bibliothèque performante open-source et disponible en français différente de celle de Google. Certaines ont été trouvé comme CMU Sphinx mais étant basé sur des modèles acoustiques anglais, les performances ne sont pas satisfaisantes. A ce jour aucune API viable différente de celle de Google n'a été trouvé et la recherche est toujours en cours

RECUPERATION DES INFORMATIONS

En termes de récupération d'information en ligne, les pistes les plus prometteuses sont les flux RSS et les données d'un serveur pour récupérer des données temporelles par exemple.

IMPLEMENTATION

STRUCTURATION DE LA SOLUTION



DICTIONNAIRE DES FONCTIONS

Signature de la fonction	Fichiers	Description
debutTransmission()	Interactions.py	Procédure qui déclenche un signal sonore pour signaler le début du processus de reconnaissance vocale.

finTransmission()		Procédure qui déclenche un signal sonore pour signaler la fin du processus de reconnaissance vocale.
Repondre(reponse)		Permet à l'ordinateur de lire un texte : module TTS utilisant l'API Google
RepondreESpeak(texte,nomFichier)		Permet à l'ordinateur de créer un fichier audio de l'ordinateur qui lit un texte entré en paramètre.
RepondreESpeakStreaming(texte)		Permet à l'ordinateur de lire un texte : module TTS utilisant l'API ESpeak
CreationBaseDeConversation(fichierDialogue)		Permet de créer une base de données de question-réponse pour le ChatBot à partir d'un fichier texte contenant des questions réponses.
getVDM()		Permet d'obtenir une « Vie De Merde » recueil populaire d'histoires drôles, à partir du flux RSS.
DonneLHeure()	dateEtHeure.py	Permet au logiciel de renvoyer l'heure de manière formatée c'est-à-dire prête à être dite par le logiciel
DonneLaDate()		Permet au logiciel de renvoyer la date de manière formatée c'est-à-dire prête à être dite par le logiciel
getDerniersMatchs()	discussionFoot.py	Permet d'obtenir les derniers matchs de ligue 1 accompagné de leur score.

REVUE DE CODE SUR LES POINTS PARTICULIERS

DESCRIPTION GLOBALE

AVANT-PROPOS

Le système réagit des stimuli oraux de l'utilisateur. Il a donc en résumé deux manières de réagir. Si le robot est face à une interaction sans intérêt particulier (« Bonjour », « J'ai faim »…) ou qu'il ne peut pas traiter tout seul (« J'ai très mal » par exemple), le logiciel optera donc pour une réponse toute faite & systématique (d'où a présence d'un ChatBot) : Typiquement « J'ai mal » entrainera la réponse : « Appelle l'infirmière ». L'autre cas est celui de demandes particulières (typiquement une demande d'information assimilable à ce qu'on pourra faire avec un moteur de recherche. Le logiciel déclenche alors un script qui lui permet de répondre à la requête de l'enfant.

PSEUDO-CODE

INITIALISATION

Initialisation des instances

ALGORITHME

Départ de la session :

Requete= ReconnaissanceVocale()

Si la Requete contient le terme « heure » :

Renvoie l'heure

Si la Requete contient le terme « date » :

Renvoie la date

Si la Requete contient « blague » :

Renvoie une blague

Si la Requete contient « parle-moi »

Renvoie une conversation sur la ligue 1 de Football

Si la Requete contient un message contenu dans la BDD du ChatBot

Renvoie la réponse associée dans la BDD

Si la Requete n'a pas de réponse associée :

Renvoie la réponse par défaut

Si la Requete contient « Fin de la discussion »

Sort de la boucle et ferme le programme

Fin de la session

INTERACTIONS ROBOT-UTILISATEUR

Tous les mécanismes d'interactions sont réunis dans le script *interactions.py* et regroupe donc STT, TTS, et fonctions relatives au ChatterBot : Ces mécanismes constituant le cœur du projet, on s'attachera dans cette partie à détailler leur fonctionnement à travers la description détaillées des procédures :

MODULE STT : COMPREHENSION DE L'UTILISATEUR

```
def ReconnaissanceVocaleGoogle():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        #print("Dites quelques chose")
        audio = r.listen(source)
    try:
        return(r.recognize_google(audio,language='fr'))
    except sr.UnknownValueError:
        print("Cherry reflechit attend le signal sonore avant de parler")
    except sr.RequestError as e:
        print("Le serveur a invalide votre requete; {0}".format(e))
```

Le module de reconnaissance vocale est assuré par l'API Google. La démarche consiste donc à créer une nouvelle instance de reconnaissance vocale et d'essayer de reconnaître ce que dit l'utilisateur en désignant comme source le microphone.

A cette fonction, on ajoute la gestion de plusieurs exceptions potentielles à savoir que l'utilisateur n'est pas compris (erreur interne au programme) ou que le serveur échoue (erreur externe au programme).

MODULE TTS : REACTION DU ROBOT

```
def Repondre(reponse):
        pygame.mixer.init()
        tts=gTTS(reponse,lang='fr')
        tts.save("temp.mp3")
        pygame.mixer.music.load("temp.mp3")
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
        os.remove("temp.mp3")
```

Le module de réponse choisi actuellement utilise l'API Google : Le principe général est de créer un fichier .mp3 qui contient le message du robot et de le lire. On supprime le fichier à la fin de la diffusion afin de ne pas surcharger la mémoire du robot et se rapprocher d'une technique de streaming.

Un bon candidat qui fonctionne en streaming est l'API ESpeak :

```
def RepondreESpeakStreaming(texte):
   subprocess.call(["espeak",texte,"-vfr"])
```

La fonction est de façon évidente plutôt simple, la commande adaptée pour le projet existant directement dans la librairie.

CHATBOT & REACTIONS PARTICULIERES

Le « ChatBot » en tant que fonction peut être divisée en deux parties : Une partie pour les réactions primaires (« Bonjour » >> « Bonjour toi »), géré par un chatBot basique, et une partie pour des réactions spécifiques c'est-à-dire qu'un appel de cette fonction déclenche des scripts spécifiques, géré par une boucle sur l'algorithme principal. (« parle-moi » >> *discussionFoot.py*)

DEMARRAGE DU CHATBOT

```
chatbot =
ChatBot('Cherry',storage_adapter='chatterbot.storage.JsonFileStorageAdapter',logic_adapter
s=[{'import_path':'chatterbot.logic.BestMatch'},{'import_path':'chatterbot.logic.LowConfid
enceAdapter','threshold':0.6,'default_reponse':'Je ne sais
pas'}],trainer='chatterbot.trainers.ListTrainer')
chatbot.set_trainer(ListTrainer)
dialogue=CreationBaseDeConversation('dialogue_general.txt')
chatbot.train(dialogue)
```

On initialise le ChatBot conformément aux constructeurs de la libraire Python ChatterBot.

La fonction train(*fichier*) permet au logiciel de créer un fichier JSON afin de sérialiser la réponse. D'un point de vue machine, les données sérialisées sont désérialisées et mise en mémoire sous forme d'une liste.

CHOIX D'UNE REPONSE A PARTIR DU CHATBOT

```
            if str(chatbot.get_response(requete))=="I'm sorry, I do not understand.":
                Repondre('Je ne comprend pas la demande')
                print("Je ne comprend pas la demande")
            else:
                Repondre(str(chatbot.get_response(requete)))
                print(str(chatbot.get_response(requete)))
```

La fonction get_reponse(*requete)* cherche si dans la liste si la valeur de *requete* existe et renvoie la valeur de l'élément suivant de la liste.

JUSTIFICATION DES CHOIX DE CONCEPTION

CHOIX DE L'API GOOGLE

L'API Google, initialement théoriquement présente dans le projet, a été choisi par les chefs de projet comme celle étant la plus viable. Cependant, rien n'étant implémenté la liberté a été prise d'explorer d'autres solutions car elle comporte des désavantages indéniables comme l'impossibilité à partir de python de faire de la reconnaissance vocale en streaming comme le ferait d'autres API, ce qui rend l'API Espeak beaucoup plus réactive que son homologue de chez Google. Cependant, le côté mécanique de l'API Espeak rend le message moins compréhensible qu'avec l'API Google.

On a trouvé suite à l'état de l'art deux API viable : Celle de Google et Espeak. On comparera donc par la suite les deux. Les deux versions ont été implémenté à travers les fichiers *cherry.py* et *cherry_espeak.py.* Sur une simple requête, typiquement « Bonjour » puis « Quel jour sommes-nous ? » et « merci » on observera le temps nécéssaire à l'opération en ne prenant pas en compte les échecs de reconnaissance, ces derniers étant lié à l'API Google.

CREATION D'UN FICHIER UTILISATEUR.PY

Ce fichier est créé dans le but d'être continué afin de pouvoir stocker des données propre à l'utilisateur vouées à être utilisé pour paramétrer le robot en début de discussion.

RECETTES

TESTS UNITAIRES

Id	Nom	Description	Validation
TU_1	Réponse par défaut	Boucle qui simule la réponse par défaut du ChatBot en cas de non détection des stimuli	*Validé*
TU_2	Réponse ChatBot réponse standard	Le but est de tester le ChatBot permettant de renvoyer des réponses standards c'est-à-dire ne nécessitant l'utilisation d'un script	*Validé*

TESTS DE VALIDATION

Id	Nom	Description	Validation
TV_1	Fermeture de la conversation	Ce test a pour but d'éteindre le programme en utilisant la commande « fin de la discussion »	*Validé*
TV_2	Heure	Ce test doit permettre de confirmer qu'il est possible pour le logiciel de donner l'heure de façon intelligible.	*Validé*
TV_3	Date	Ce test doit permettre de confirmer qu'il est possible pour le logiciel de donner la date du jour de façon intelligible.	*Validé*
TV_4	Derniers matchs ligue 1	Ce test doit confirmer que le logiciel peut donner les deux derniers matchs de ligue 1.	*Validé*
TV_5	Blague	Ce test doit permettre de confirmer qu'il est possible pour le logiciel de faire une blague	*Validé*
TV_6	Réponse par défaut	Le but est de tester qu'en cas de non gestion de la réponse, le robot sort une réponse par défaut.	*Validé*
TV_7	Réponse banale	Le but est de vérifier si le chatBot répond bien ce qu'on lui a dit de répondre.	*Validé*

GESTION DE PROJET

EXIGENCES

EXIGENCES FONCTIONNELLES

EF_1 : La solution devra pouvoir produire un résultat oral.

EF_11 : La solution devra pouvoir produire un résultat en français.

EF_12 : La solution devra pouvoir produire un résultat compréhensible.

EF_2 : La solution devra être capable de comprendre certaines phrases de l'utilisateur.

EF_21 : La solution devra pouvoir transcrire la parole de l'utilisateur en texte.

EF_22 : La solution devra comprendre le français.

EF_3 : La solution pourra répondre à certaines remarques de l'utilisateur

EF_31 : La solution devra pouvoir réagir à des remarques simples

EF_32 : La solution devra pouvoir accéder à des demandes demandant un traitement particulier (soit une synthèse d'information, soit le résultat d'un algorithme particulier.

EXIGENCES NON-FONCTIONNELLES

ENF_1 : La solution devra être compatible avec un OS Ubuntu.

ENF_2 : La solution devra être développée en se basant sur des scénarios d'utilisation.

ANALYSE DES BESOINS UTILISATEURS

PROFIL DE L'UTILISATEUR

L'utilisateur final est un enfant dont l'âge se situe entre 10 et 14 ans. De plus, la posture physique de l'enfant par rapport au robot est supposée idéale c'est-à-dire que le micro qui sera ajouté au robot lors de l'intégration de l'algorithme dans le projet final, est à une distance permettant de bien percevoir la voix de l'enfant (au niveau de la tête de l'enfant et à un mètre de sa tête).

BESOIN EN TERMES D'INTERFACE HOMME-SYSTEME

Sur ce projet, l'interfaçage entre l'enfant et le robot se fait via la communication verbale. Il est donc nécessaire de porter une attention particulière à la perception de la voix et sur la manière de mener la conversation. Aussi, un des enjeux est de permettre une discussion entre le système et l'utilisateur ou du moins une réaction orale du robot face à un stimulus de l'utilisateur.

Fig. : Image d'un cas concret d'utilisation du Robot Cherry

MATRICE DES RISQUES

MATRICES DES RISQUES TECHNIQUES & FACTEUR HUMAIN

ID	Description	Probabilité (entre 0 et 1)	Criticité (entre 1 et 5)	Solution envisagée
RTH1	La connexion est de trop basse qualité pour assurer un traitement correct des requêtes du client vers les serveurs gérant la synthèse vocale.	0.4	5	Trouver un moyen de changer la connexion, en utilisant pas un Wifi mais une clé 3G, ou autre système permettant une connexion correcte.

RTH2	Le bruit ambiant est trop important	0.2	3	L'idée serait soit de mettre un filtre en aval du microphone, soit de proposer la solution uniquement dans un cadre relativement silencieux.
RTH3	L'enfant ne comprend pas ce que dit le robot	0.05	4	Il est alors envisagé de changer légèrement la voix via le paramétrage de l'API.
RTH4	L'enfant n'arrive pas à interagir avec le robot du fait de formulation particulière à l'enfant	0.1	3	L'idée serait alors de remodeler les requêtes en fonction de son vocabulaire et de sa façon de parler quitte à faire des commandes personnalisées
RTH5	La carte Odroid ne suffit pas à prendre en charge le programme.	0.01	5	L'idée est de changer la carte Odroid par un microcontrôleur plus puissant comme un Raspberry P (démarche en cours dans le projet)

MATRICES DES RISQUES GESTION DE PROJET

ID	Description	Probabilité (entre 0 et 1)	Criticité (entre 1 et 5)	Solution envisagée
RGESP1	Le travail étant conséquent, un des risques de ne pas assez se focaliser pour aboutir à un résultat.	0.7	5	La solution mise en place lors de ce projet est de se focaliser sur un scénario d'utilisation.

PLANIFICATION

DEROULEMENT GLOBAL

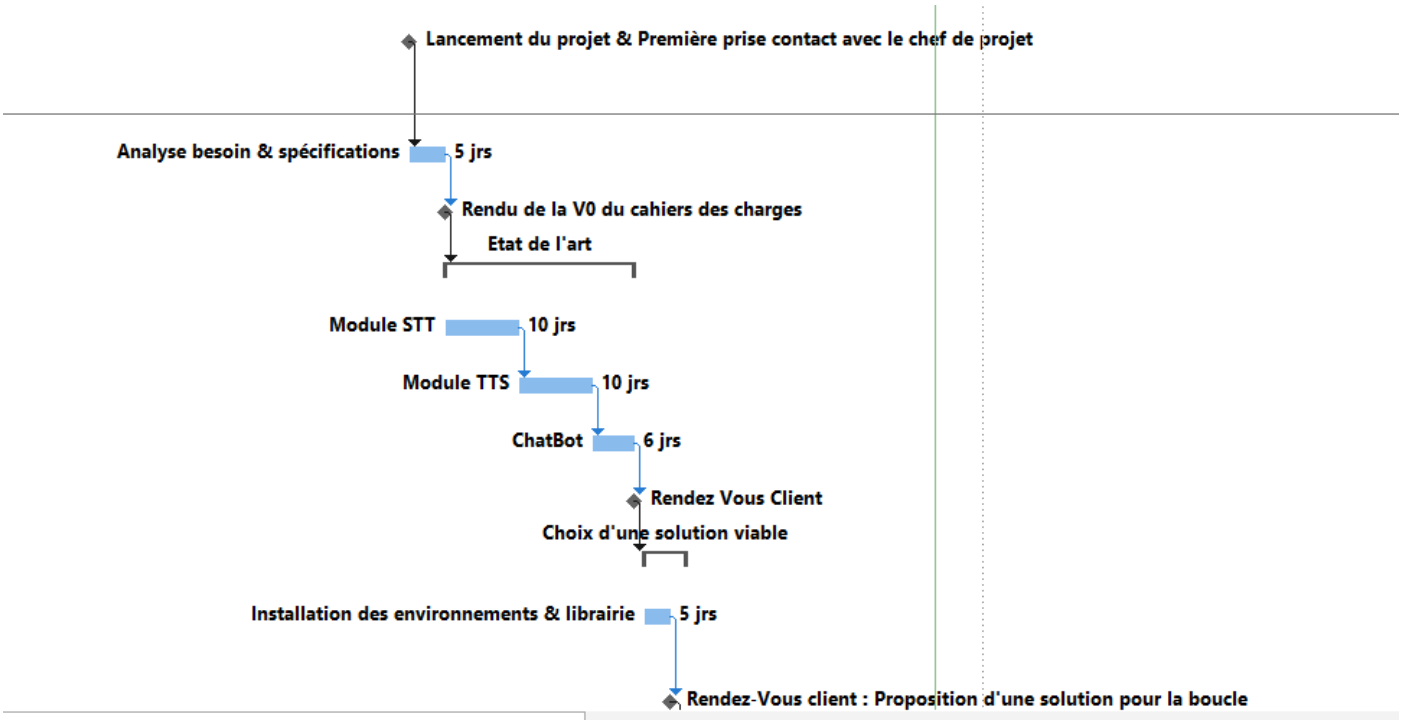
DIAGRAMME DE GANTT

LISTE DES TACHES

Nom de la tâche	Durée	Début	Fin
Lancement du projet & Première prise contact avec le chef de projet	0 jr	Ven 13/01/17	Ven 13/01/17
Analyse besoin & spécifications	5 jrs	Ven 13/01/17	Jeu 19/01/17
Rendu de la V0 du cahiers des charges	0 jr	Jeu 19/01/17	Jeu 19/01/17
Etat de l'art	**26 jrs**	**Ven 20/01/17**	**Ven 24/02/17**
Module STT	10 jrs	Ven 20/01/17	Jeu 02/02/17
Module TTS	10 jrs	Ven 03/02/17	Jeu 16/02/17
ChatBot	6 jrs	Ven 17/02/17	Ven 24/02/17
Rendez Vous Client	0 jr	Ven 24/02/17	Ven 24/02/17
Choix d'une solution viable	**6 jrs**	**Lun 27/02/17**	**Lun 06/03/17**
Installation des environnements & librairie	5 jrs	Lun 27/02/17	Ven 03/03/17
Rendez-Vous client : Proposition d'une solution pour la boucle	0 jr	Ven 03/03/17	Ven 03/03/17
Mise en place d'un scénario d'utilisation	1 jr	Lun 06/03/17	Lun 06/03/17

Implémentation des fonctions	24 jrs	Mar 07/03/17	Ven 07/04/17
"Raconte moi une blague"	3 jrs	Mar 07/03/17	Jeu 09/03/17
"Donne moi l'heure"	3 jrs	Ven 10/03/17	Mar 14/03/17
"Donne moi la date"	2 jrs	Mer 15/03/17	Jeu 16/03/17
"Parlons"	5 jrs	Ven 17/03/17	Jeu 23/03/17
Mise en place des réponses standardisées : Chatbot	5 jrs	Ven 24/03/17	Jeu 30/03/17
Gestion des erreurs & exceptions	6 jrs	Ven 31/03/17	Ven 07/04/17
Test, Optimisation & réalisation des livrables	10 jrs	Lun 10/04/17	Ven 21/04/17
Rendu du projet	0 jr	Jeu 27/04/17	Jeu 27/04/17
Soutenance finale	0 jr	Mar 02/05/17	Mar 02/05/17

DIAGRAMME



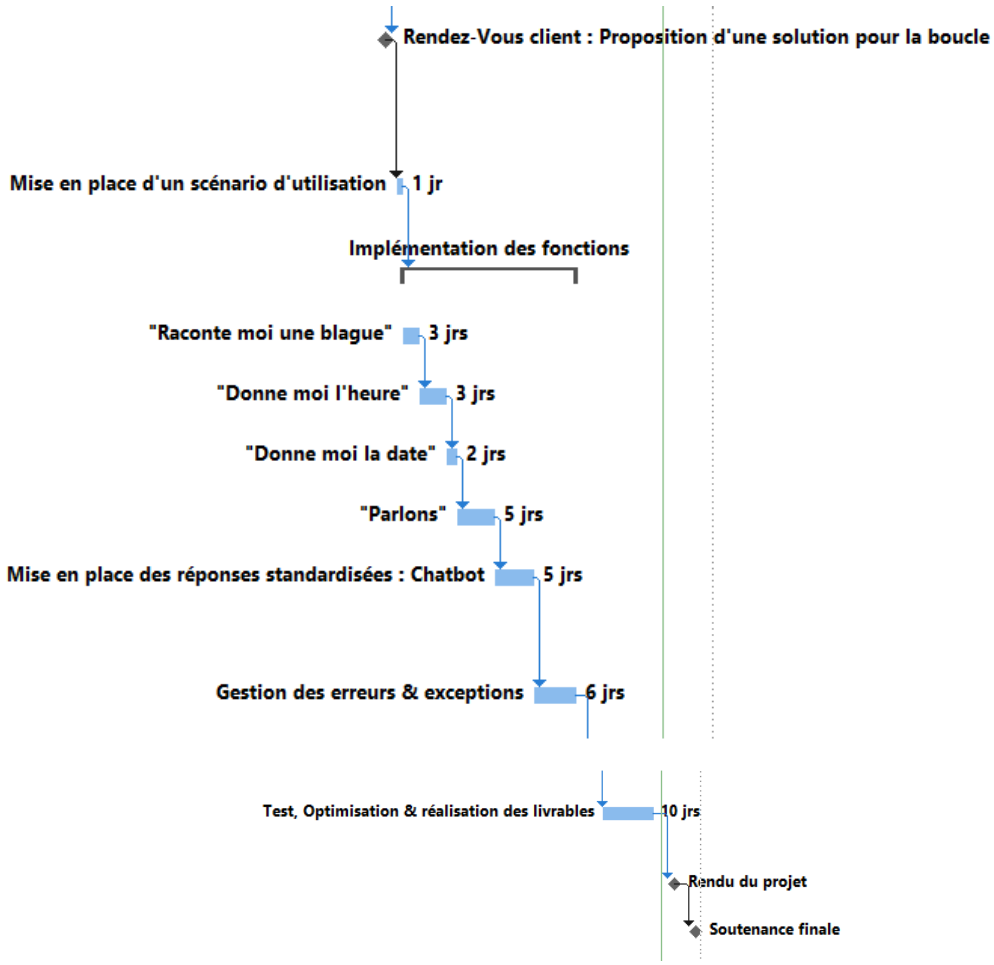


ILLUSTRATION DU SYSTEME PAR UN CAS D'UTILISATION

MISE EN PLACE DES PERIPHERIQUES TECHNIQUES (PROPRE AU PROJET)

INSTALLATION DU PROGRAMME

Dans un premier temps, il est nécessaire d'installer toutes les dépendances nécessaires au logiciel. Un script shell appelé *install.sh* est prévu à cet effet. Les commandes nécessitant les droits d'administrateur, il faut déclencher le script avec les droits d'administrateur. (commande *sudo* sur un OS Linux)

CAS D'UTILISATION

L'essentiel des interactions sont orales, il est donc clair que les images suivantes ne permettent pas de représenter la totalité du programme. Cependant, pour les développeurs, des retours de la console permettent d'avoir une idée du fonctionnement. C'est donc les retours de la console que l'on présentera dans cette partie :

Dans le cadre du scénario d'utilisation, Pierre voit le robot s'allumer, il tente alors de lui parler et voici le retour console d'un point de vue développeur.

```
gregoire@ubuntu:~/Desktop/Soutenance/Cherry_project/cherryProject/verbalCommunication/cherryProject$ sudo python cherry.py
/usr/local/lib/python2.7/dist-packages/chatterbot/storage/jsonfile.py:26: UnsuitableForProductionWarning: The JsonFileStorageAdapter is not recommended for production environments.
```

```
  self.UnsuitableForProductionWarning
ALSA lib pcm_dmix.c:1029:(snd_pcm_dmix_open) unable to open slave
ALSA lib setup.c:548:(add_elem) Cannot obtain info for CTL elem (MIXER,'AC97 2ch->4ch Copy Switch',0,0,0): No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround41
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround50
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround51
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround71
ALSA lib setup.c:548:(add_elem) Cannot obtain info for CTL elem (PCM,'IEC958 Playback PCM Stream',0,0,0): No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm_dmix.c:1029:(snd_pcm_dmix_open) unable to open slave
Tu as dit :
Bonjour
bien le bonjour lami
Tu as dit :
qui es-tu
Je suis Cherry, robot mis a disposition des enfants pour passer le temps. Mon concepteur est l'Association Prima
Tu as dit :
quel jour sommes-nous s'il te plaît
Cherry :
Nous sommes le mercredi 26 mai 2017
```

A la ligne suivante, une exception est interceptée par le logiciel qui n'a pas entendu ce que l'utilisateur dit.

```
Cherry reflechit attend le signal sonore avant de parler
Cherry reflechit attend le signal sonore avant de parler
Tu as dit :
mais au fait quelle heure il est
Cherry :
Il est 11 heures 30
Tu as dit :
```

```
raconte-moi une blague
Cherry :
Aujourd'hui, je suis animateur dans un centre de loisirs. Les ados étant très dispersés devant le guide, j'en prends un à part pour le gronder sévèrement. C'était le guide.
Cherry reflechit attend le signal sonore avant de parler
Tu as dit :
parle-moi
Cherry :
Aimes tu le football ?
Tu as dit :
oui
Cherry :
Je peux te dire les deux derniers matchs de ligue 1 qui ont eu lieu, les veux tu ?
Tu as dit :
oui
Cherry :
Les 2 derniers matchs qui ont eu lieu sont Lyon - Monaco (score final : 1-2) et Saint-Étienne - Rennes (score final : 1-1)
Cherry reflechit attend le signal sonore avant de parler
Tu as dit :
j'ai mal j'ai mal
Cherry :
appelle linfirmiere
Tu as dit :
je suis fatigué
Cherry :
Alors tu peux dormir
Tu as dit :
oui d'accord merci
Cherry :
```

A la ligne suivante, le robot détecte une expression pour laquelle il n'a pas de réponse implémentée.

```
Je ne comprend pas la demande
Cherry reflechit attend le signal sonore avant de parler
Tu as dit :
Merci
Cherry :
de rien
Tu as dit :
au sens de la discussion merci beaucoup
```

```
Cherry :
Je ne comprend pas la demande
Tu as dit :
fin de la discussion
Cherry :
Merci de la dicussion
Au revoir
```

VIDEO DU CAS D'UTILISATION

A ce stade du rapport, je vous invite à aller voir la vidéo du test de validation du scénario jointe au dossier, cette vidéo étant surement plus représentative que les retours console.

CONCLUSION

DIFFICULTES RENCONTREES

DIFFICULTES TECHNIQUES

QUALITE DE LA CONNEXION & PERFORMANCES

Comme dit en début de rapport, les performances du logiciel dépendent :

- De la qualité de la connexion,
- Des performances propres de la machine qui le fait tourner.

Ceci peut expliquer les temps d'attente lors des interactions.

UTILISATION DU MICROPHONE

Les performances du logiciel dépendent en partie de la qualité de l'information vocale qu'il perçoit. Ainsi, pour une utilisation optimale, il est conseillé d'utiliser un micro correcte ou fait pour cela comme un kit main libre par exemple.

TROUVER UNE API PERMETTANT UNE RECONNAISSANCE VOCALE EN FRANÇAIS

Une des difficultés importantes a été de trouver une API de reconnaissance vocale. Outre le choix évident de prendre celle de Google, plusieurs API gratuite existent. Cependant, les modèles acoustiques utilisés pour la reconnaissance sont basés sur des modèles anglo-saxons impliquant des performances moins importantes du fait des différences des langues (et notamment la présence d'accents en français).

TROUVER DES RESSOURCES EN LIGNE SUR LA LIGUE 1

Une des difficultés qui explique pourquoi l'on a pas pu implémenté la totalité des recommandations des fans de football (cf Partie : Résultat interview) est le fait qu'il n'existe pas de flux RSS ou de ressources gratuites permettant d'importer dans le logiciel certaines informations. En effet, la plupart des flux RSS se focalise sur des articles de journaux.

DIFFICULTES GESTION DE PROJET

CLARTE DES OBJECTIFS

Les objectifs ont changé en milieu de projet. En effet, à la base, il était demandé de trouver une alternative à l'API Google, exigences qui a évolué vers l'implémentation d'un cas d'utilisation.

DOCUMENTATION DU PROJET & TRAVAUX PRECEDENTS

D'après le chef de projet, des élèves auraient déjà travaillé sur cette partie. Le problème est que ces scripts ne sont pas très avancés et quasiment impossible à trouver sur le GitHub par exemple.

POINTS POSITIFS

UN PROJET QUI PEUT ETRE REPRIS

Une attention particulière a été portée sur les possibilités de continuer le code. En effet, les délais pour l'objectif étant relativement court il est impossible de créer un module permettant de parler et réagir comme un humain. Ainsi, le code a été fait de façon à ce qu'il soit continué de manière aisée sans avoir à tout refaire.

BASE SOLIDE & ADAPTABLE

On a une base solide pour implémenter la suite des scénarios.

PERSPECTIVES D'EVOLUTION

INTERACTIONS HOMME-ROBOT

Afin de pouvoir savoir quand parler, il a été mis en place un système d'indicateurs sonores. Cependant, il serait plus approprié de faire appel au langage corporel qu'on sait important dans la communication entre hommes. Cette option est complètement envisageable via soit un mouvement des bras, soit une animation au niveau des yeux permettant de signifier que le robot est en train d'écouter. N'ayant pas eu accès au robot, il n'a pas été possible d'envisager ce cas-là dans le projet. Dans le but d'anticiper cette problématique, un dossier *VREP* (nom du logiciel de simulation du corps du robot) a été implémenté afin de pouvoir de manière rapide simuler le robot.

CHATBOT A DEUX VITESSES

La prochaine chose à faire serait de consolider la base fournie, en implémentant un deuxième Chatbot sur le principe du premier mais qui renverrai les adresses des scripts afin de pouvoir les déclencher rendant ainsi le code moins lourd.

IMPLEMENTATIONS PROCHAINES PAR UNE CONCEPTION CENTREE UTILISATEUR

Pour une implémentation prochaine, l'idée serait d'aller voir les utilisateurs du robot (les enfants hospitalisés) afin de faire des tests des interactions qu'il désirerait avoir et qu'ils auront. En effet, les enfants s'expriment différemment des adultes et il serait intéressant de pouvoir mener des entretiens afin d'avoir une base de données solides d'interactions potentielles. En d'autres termes, l'interview des enfants permettrait une modélisation des interactions qu'il pourrait avoir avec le robot Cherry, ce qui pourrait être une alternative à l'implémentation sérielle de scénarios d'utilisation.

ANNEXES

- Code Source
- Livrable Intermédiaire
- DataSheet
- Vidéo du test de validation du scénario d'utilisation