PPE - RoboCup@Home - Dialogue

Ecole Nationale d'Ingénieurs de Brest - 2022

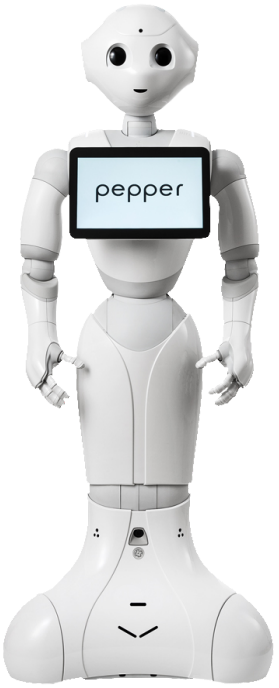
Les encadrants:
M. BUCHE, Cédric - M. DESMEULLES, Gireg et M. NEAU, Maëlic

Les auteurs:
M. LOPEZ, Valentin - M. MENA, Trajano

1. Introduction

Dans la compétition RoboCup@Home a pour but de développer des robots d'aide à la personne. Ils doivent être capables de se mouvoir dans des pièces, d'interagir avec l'humain et même de manipuler des objets. Un ensemble de tests de référence est utilisé pour évaluer les capacités et les performances des robots dans un environnement domestique non standardisé réaliste. L'accent est mis sur les domaines suivants: interaction et coopération homme-robot, navigation et cartographie dans des environnements dynamiques, vision par ordinateur et reconnaissance d'objets dans des conditions de lumière naturelle, manipulation d'objets, comportements adaptatifs, intégration de comportements, intelligence ambiante, normalisation et intégration des systèmes. Pour gagner, le robot doit être capable de faire autant de tâches que possible. Pour ce faire, le robot doit interagir dans différents domaines : perception, dialogue, navigation. Ceux-ci doivent travailler ensemble.

Le robot Pepper est utilisé, il a le système opératif Naoqi.



Source: https://www.softbankrobotics.com/emea/en/pepper

Certaines de ses caractéristiques sont:

- 20 degrés de liberté pour les mouvements naturels et expressifs.
- Reconnaissance vocale et dialogue disponibles en 15 langues.
- Modules de perception pour reconnaître et interagir avec la personne qui lui parle.
- Capteurs tactiles, LED et microphones pour des interactions multimodales.
- Capteurs infrarouges, pare-chocs, une unité inertielle, caméras 2D et 3D, et sonars pour la navigation omnidirectionnelle et autonome.
- Plateforme ouverte et entièrement programmable.

Ce rapport traite de la partie dialogue. L'objectif était de produire un système capable de détecter la parole, de la transformer en texte et de la comprendre correctement. Le travail a été entièrement fait sur Python 3, et dans certains cas ROS a été utilisé pour se connecter avec les autres modules. Le ROS (Robot Operating System) est un ensemble de bibliothèques de logiciels et d'outils qui nous aident à construire des applications de robot. Il fournit des services tels que l'abstraction du hardware, le contrôle des périphériques de bas niveau, le passage de messages entre les processus et la gestion des paquets. Les ensembles de processus basés sur ROS en cours d'exécution sont représentés dans une architecture graphique où le traitement a lieu dans des nœuds qui peuvent recevoir, publier et multiplexer des données de capteurs, de contrôle, d'état, de planification, de l'actionneur et d'autres messages.

La première partie est le traitement du son, qui consiste à écouter le son, à détecter quand une personne parle et à enregistrer ce qu'elle dit. Cela peut être difficile car il peut y avoir des bruits de fond ou d'autres sons qui ne devraient pas être enregistrés (bruit parasites). Ensuite, le son enregistré doit être transformé en texte. Cela ne devrait pas nécessiter une connexion Internet parce qu'il n'y a peut-être pas de connexion stable dans la compétition. Enfin, un code du Traitement Automatique du Langage Naturel (NLP) est utilisé pour que le robot comprenne les intentions de l'utilisateur. Par exemple, si quelqu'un dit "take me the book to Valentin" à Pepper, ce système va, à la fin, retourner la chaîne : "{'intent': 'take', 'object': 'apple', 'destination': 'Valentin'}". Le robot saura alors ce qu'il a à faire. Un code question-réponse a également été développé en utilisant la base de données de connaissance de Wikipedia. Tout cela doit se faire en anglais.

2. Analyse du Signal Sonore

Cette partie se réfère à la détection des sons de l'environnement afin de détecter un interlocuteur potentiel parlant au robot et une fois qu'il est détecté, l'enregistrement de ce qu'il a dit.

En premier lieu la partie de la détection. Il y a 3 paramètres à prendre en compte :

- **Counter Speech**: c'est le nombre d'itérations qui doivent passer pour passer à l'état **Parole**.

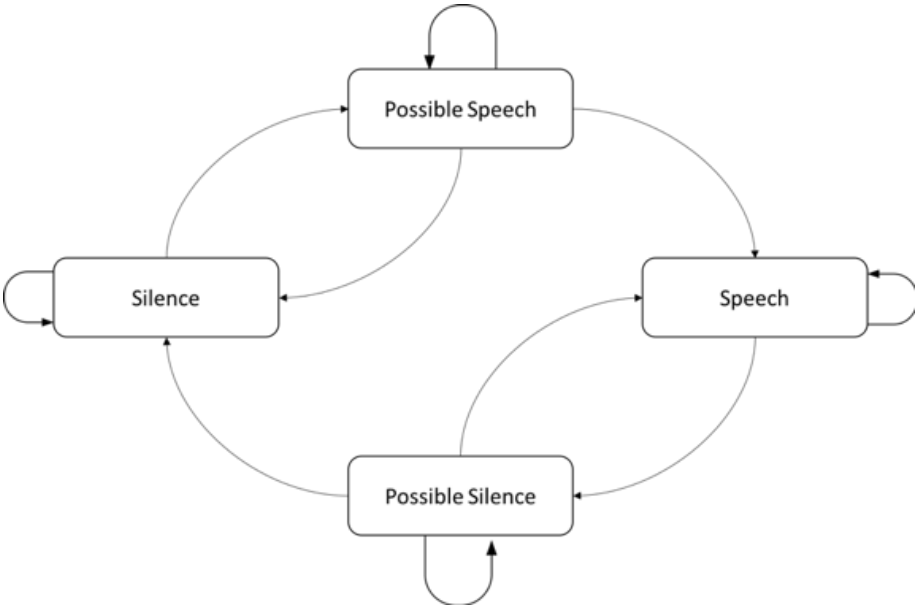
- **Threshold Offset**: c'est une valeur qui est ajoutée à la moyenne, qui est expliquée ci-dessous, pour utiliser la valeur résultante comme un seuil à franchir afin de passer à l'état **Parole Possible**.

- **Counter Silence:** c'est le nombre d'itérations qui doivent passer pour passer à l'état **Silence**.

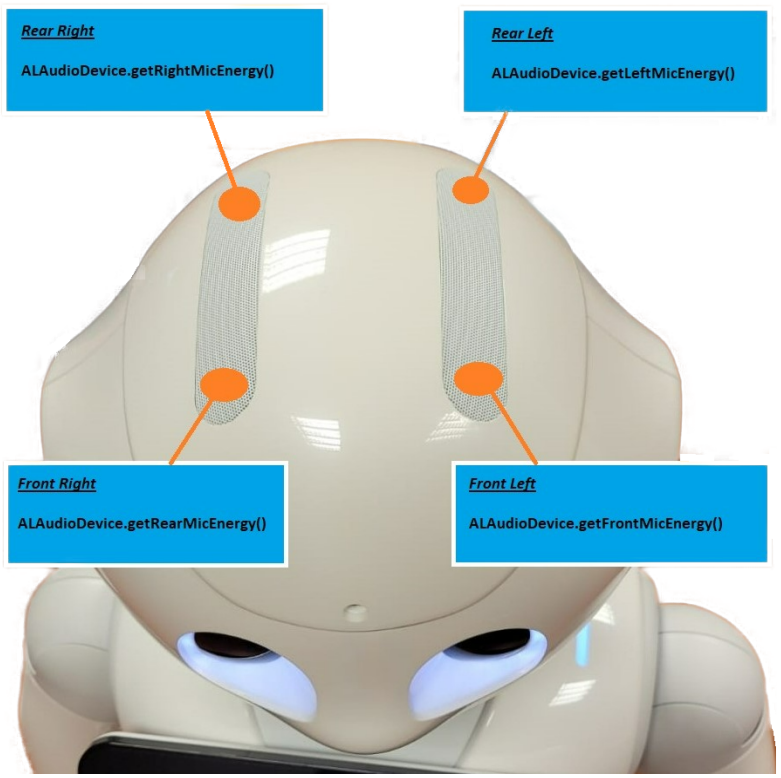
La façon dont ils sont utilisés est expliquée ci-dessous.

Tout le système d'analyse du signal sonore est une machine à état qui peut être dans l'un des 4 états suivants: **Silence**, **Parole Possible**, **Parole**, **Silence Possible**.

La manière de passer d'un état à l'autre est la suivante :



Au début, le programme démarre dans l'état Silence. A chaque itération, 3 valeurs sont calculées : l'**energy** qui est la moyenne du "niveau d'énergie" que les quatre microphones du robot enregistrent, **ymed** qui est la moyenne entre les valeurs d'énergie les plus élevées et les plus basses enregistrées au cours des quelques itérations précédentes et la **threshold** qui est utilisée pour passer de **Parole Possible** à **Parole**.



Silence → Parole Possible

Pour passer à **Parole Possible**, la valeur de l'**energy** doit dépasser la valeur **ymed (t-1)** + **Threshold Offset**. Le **threshold** est alors fixé à cette valeur et le **Counter Speech** commence à compter.

Parole Possible → Parole

Une fois dans l'état **Parole Possible**, la valeur de l'**energy** actuelle doit rester supérieure au **threshold** et au **ymed(t)**. Si cette condition est maintenue jusqu'à ce que le **Counter Speech** atteigne zéro, l'état passe à **Parole**. Sinon, l'état redevient **Silence**.

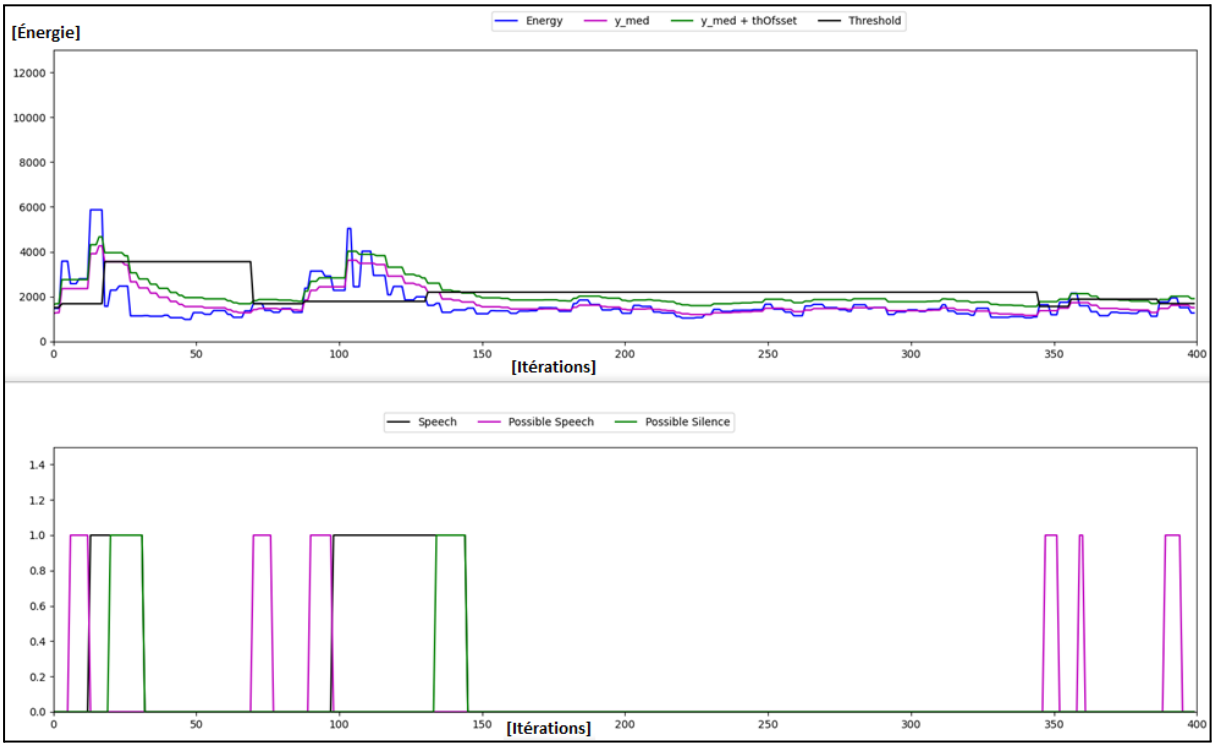
Parole → Silence Possible

Pendant l'écoute, si la valeur de l'**energy** actuelle diminue en dessous de **ymed(t)** et du **Threshold**, l'état passe à **Silence Possible**. Ensuite, le **Threshold** est fixé à **ymed(t)** et le **Counter Silence** commence à compter.

Silence Possible → Silence

Dans l'état **Silence Possible**, si la valeur repasse au-dessus du **Threshold** avant que le **Counter Silence** n'atteigne zéro, l'état revient à **Parole** et le **Counter Silence** est remis à zéro. Sinon, l'état passe à **Silence**.

Le graphique suivant montre le fonctionnement de toutes les règles expliquées ci-dessus.



Étant en état **Parole**, le robot commence à enregistrer tout, mais il y a un problème dont il faut obligatoirement tenir compte. Pour atteindre la condition précédente et commencer l'enregistrement, le locuteur doit avoir parlé avant que l'enregistrement lui-même ne

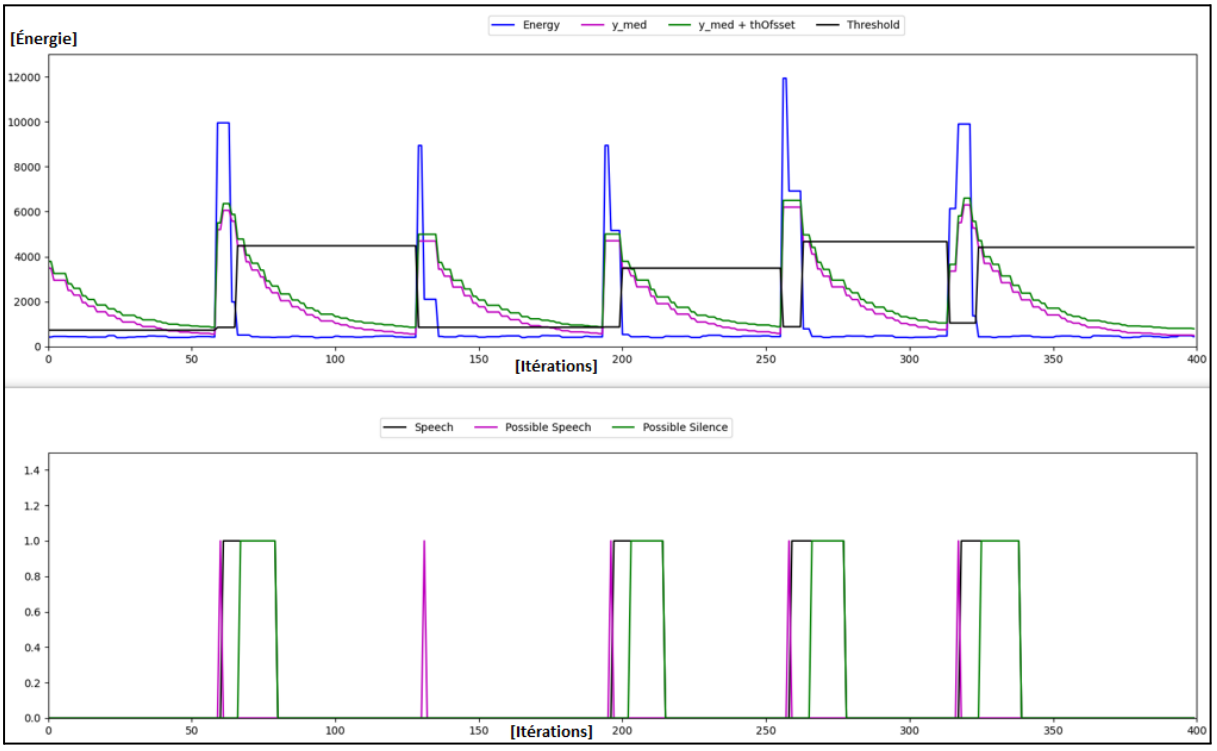
commence. Cela signifie qu'au moment où l'enregistrement a commencé, la personne avait déjà dit une partie de ce qu'elle était en train de dire, il faut donc récupérer cette information antérieure.
Afin de conserver en permanence les dernières secondes de sons captées par les microphones, nous avons enregistré dans une "**queue**" toutes les informations qui apparaissent dans le "**input buffer**" en utilisant une méthode PEPS (*Premier Entré, Premier Sorti*).
Lorsque la file d'attente est pleine, il suffit d'effacer les données les plus anciennes dans la **queue** et d'ajouter les plus récentes. De cette façon, la **queue** contient toujours les informations manquantes qui ont été mentionnées. Ensuite, lorsque l'état passe à **Silence**, l'enregistrement s'arrête et ces informations sont simplement ajoutées au début de l'enregistrement.

Les explications suivantes permettent de comprendre chacun des paramètres ci-dessus et leur influence.

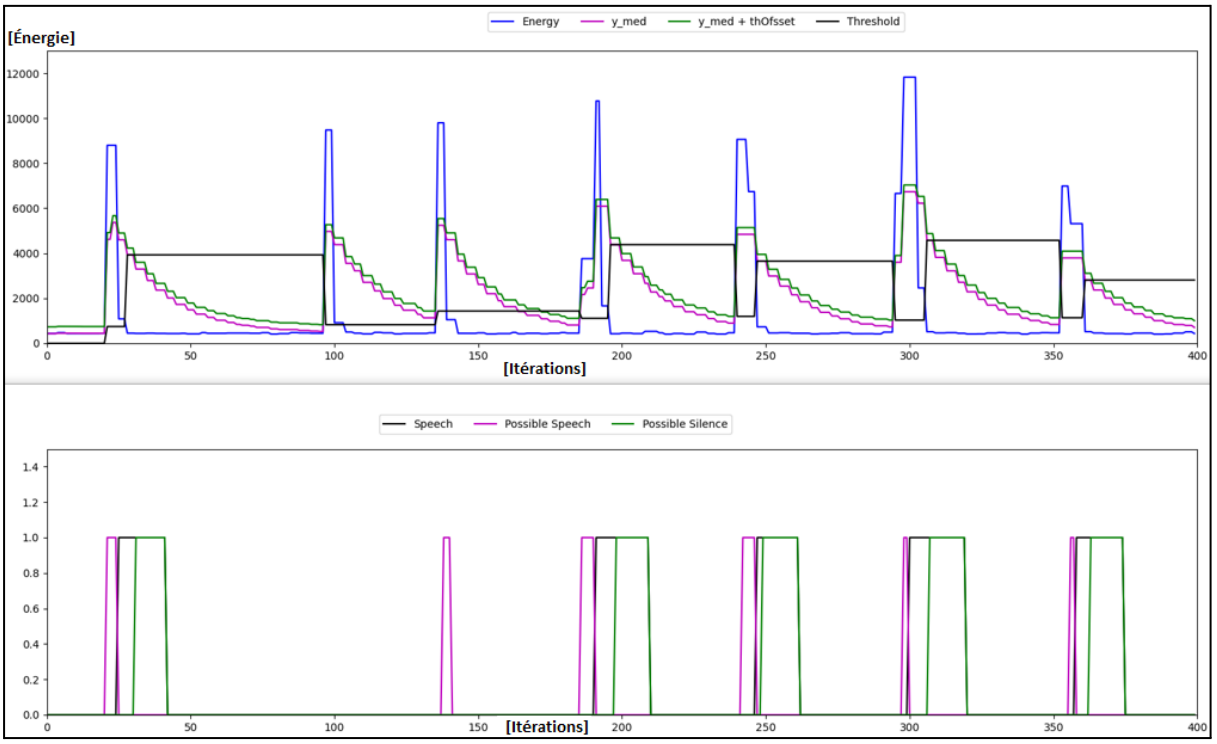
2.1. L'influence du Counter Speech

Tout d'abord, regardons ce qui se passe avec la variation de la valeur de **Counter Speech**. Quelques tests ont été effectués pour les valeurs 2, 3, 4 et 5 [itérations]. Le but était d'essayer de filtrer au mieux les bruits impulsifs dans l'environnement. Les bruits impulsifs utilisés pendant le test étaient des "coups de gobelet" et des "applaudissements".

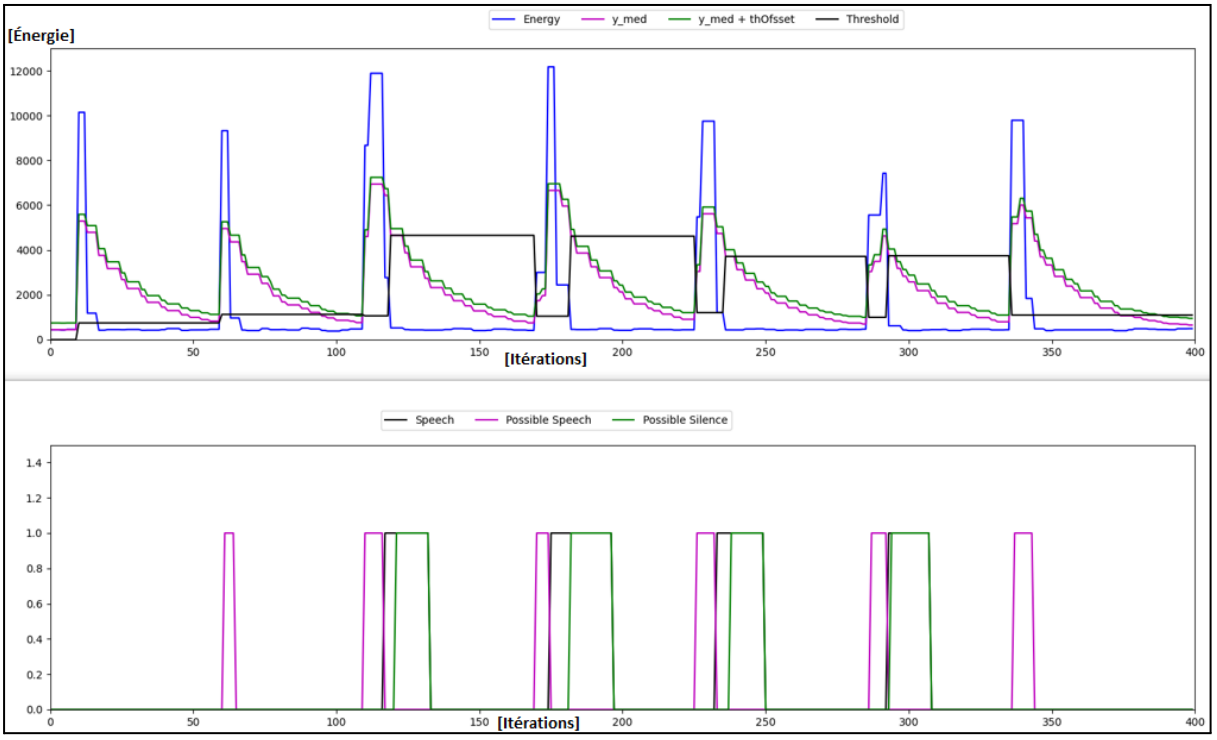
Counter Speech = 2



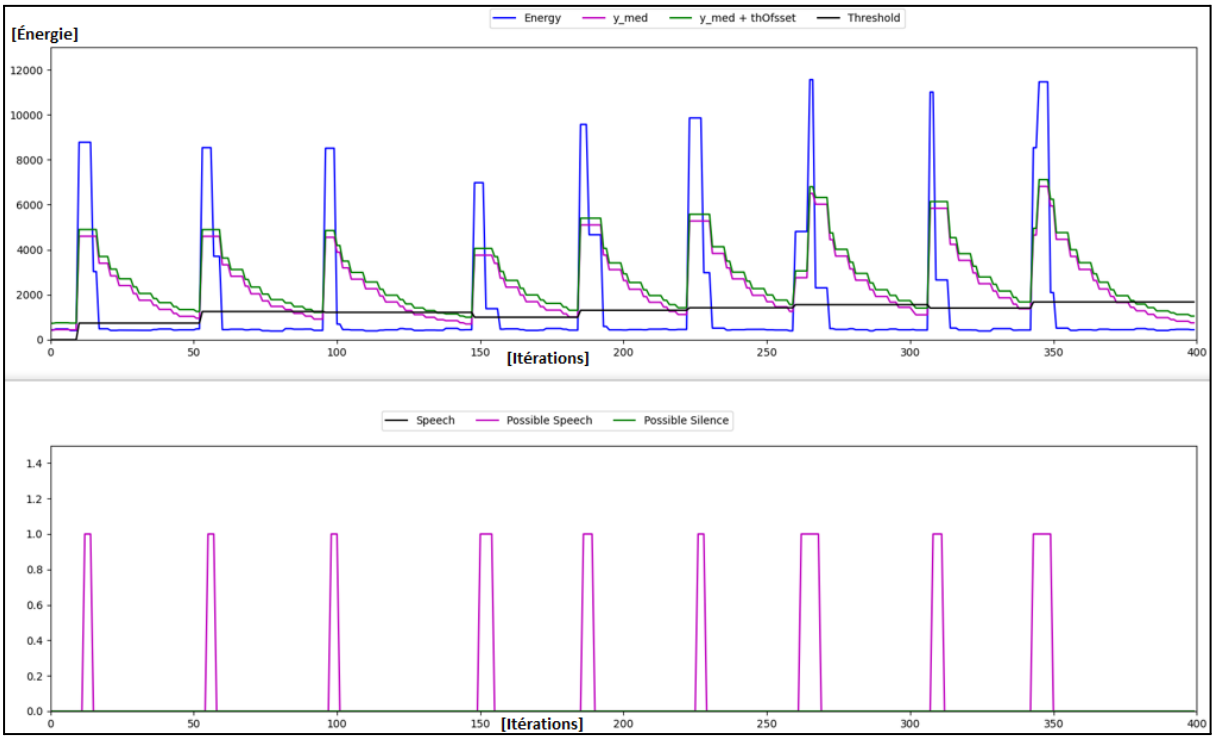
Counter Speech = 3



Counter Speech = 4



Counter Speech = 5



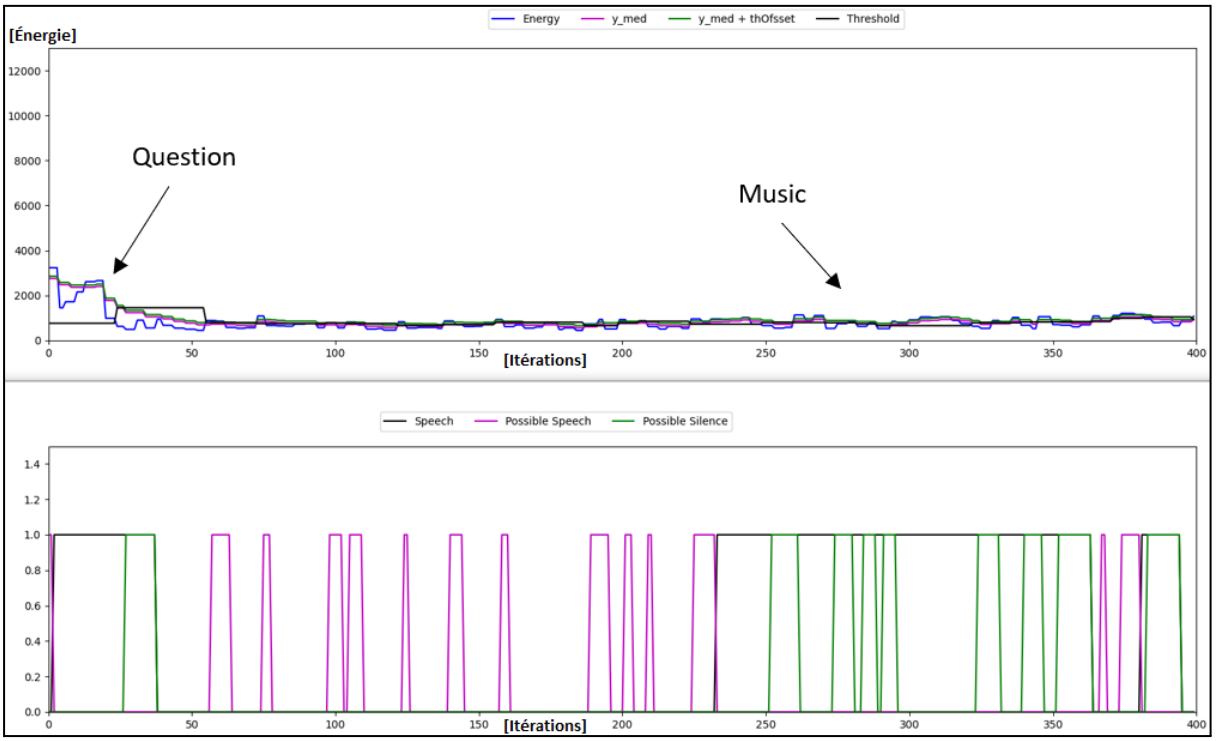
Le résultat était que plus la valeur de **Counter Speech** est élevée, meilleur est le filtrage du bruit impulsif. Comme la valeur 5 filtre tous les bruits, elle semble être le choix parfait. Mais il y a un problème qui doit être pris en compte et qui sera traité dans la section suivante.

2.2. L'influence de la Threshold Offset

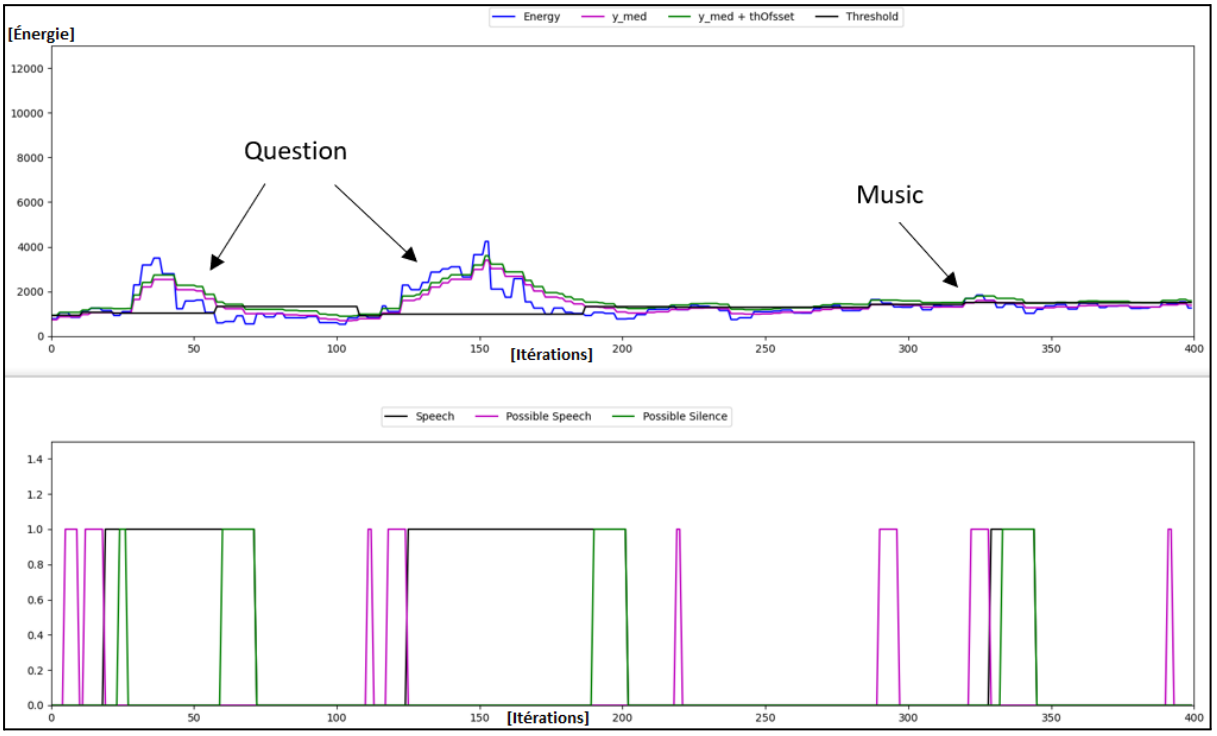
Dans ce test, nous avons posé quelques questions avec de la musique comme bruit de fond. Les valeurs de 100 à 500 $[énergie]$[1] ont été testées en utilisant 5 comme valeur de **Counter Speech**.

1 c'est l'unité d'énergie qui retourne les fonctions du système opératif Naoqi qui collecte les données des microphones du robot

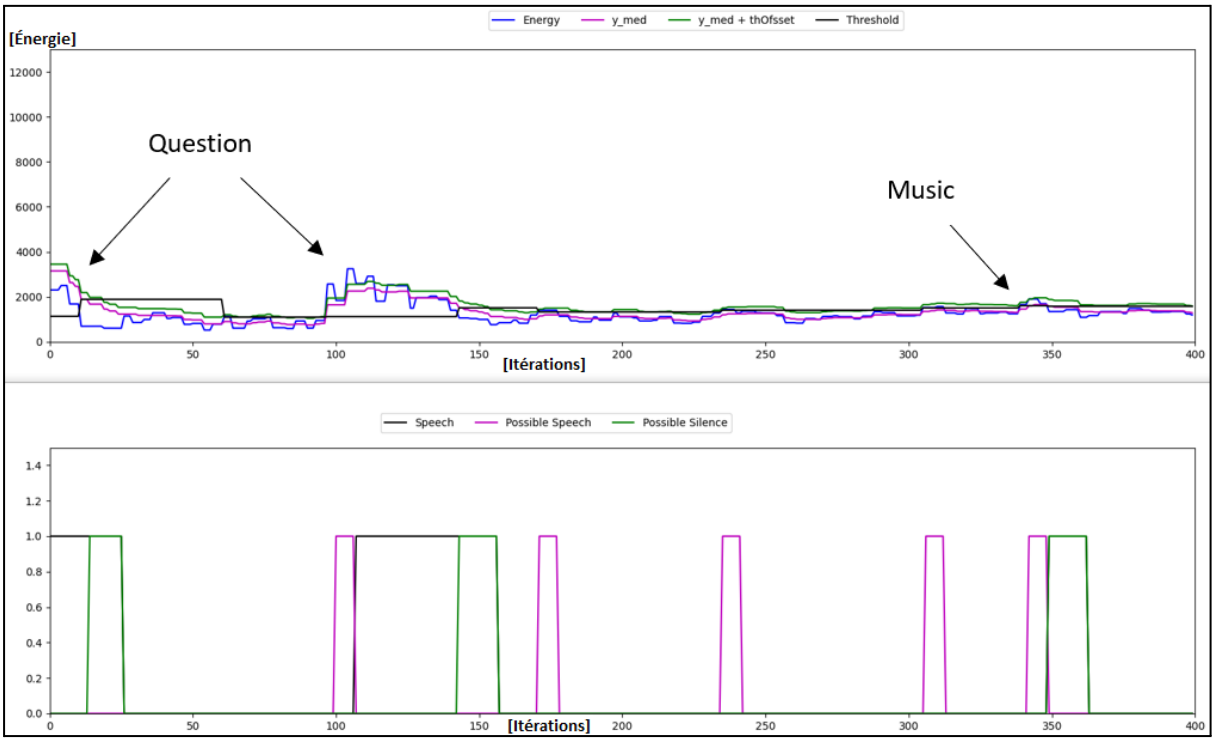
Threshold Offset = 100



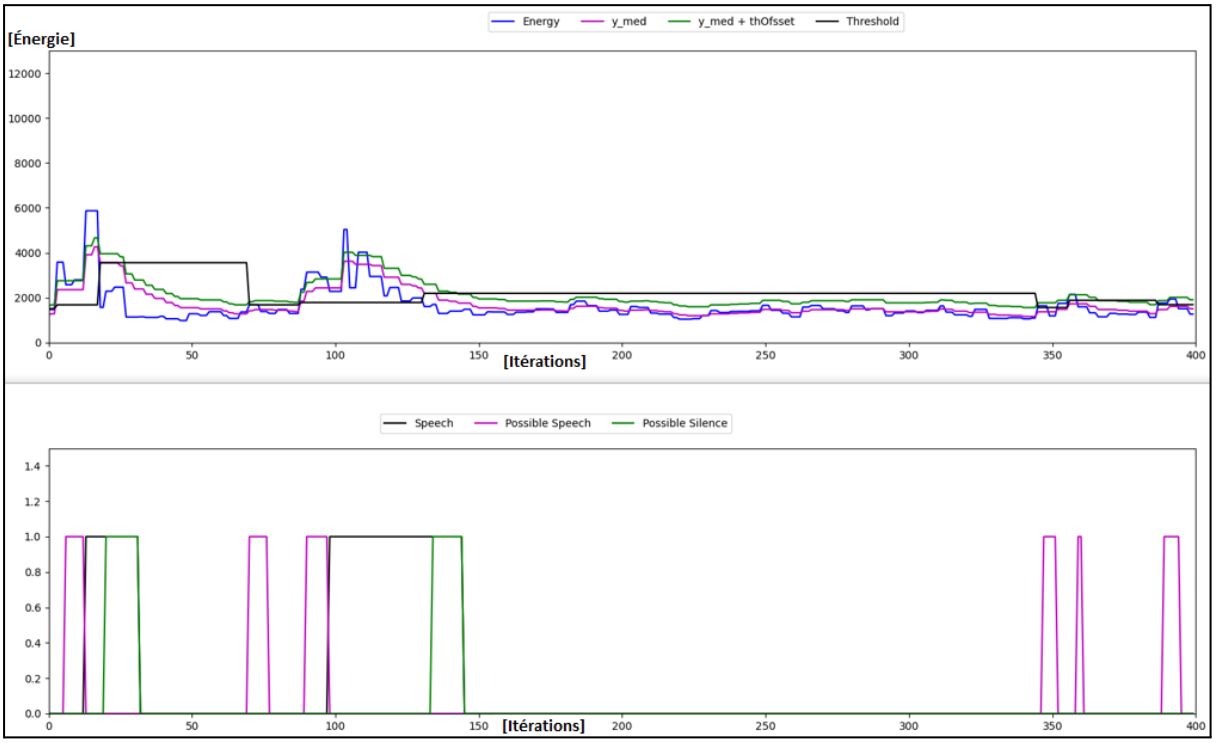
Threshold Offset = 200



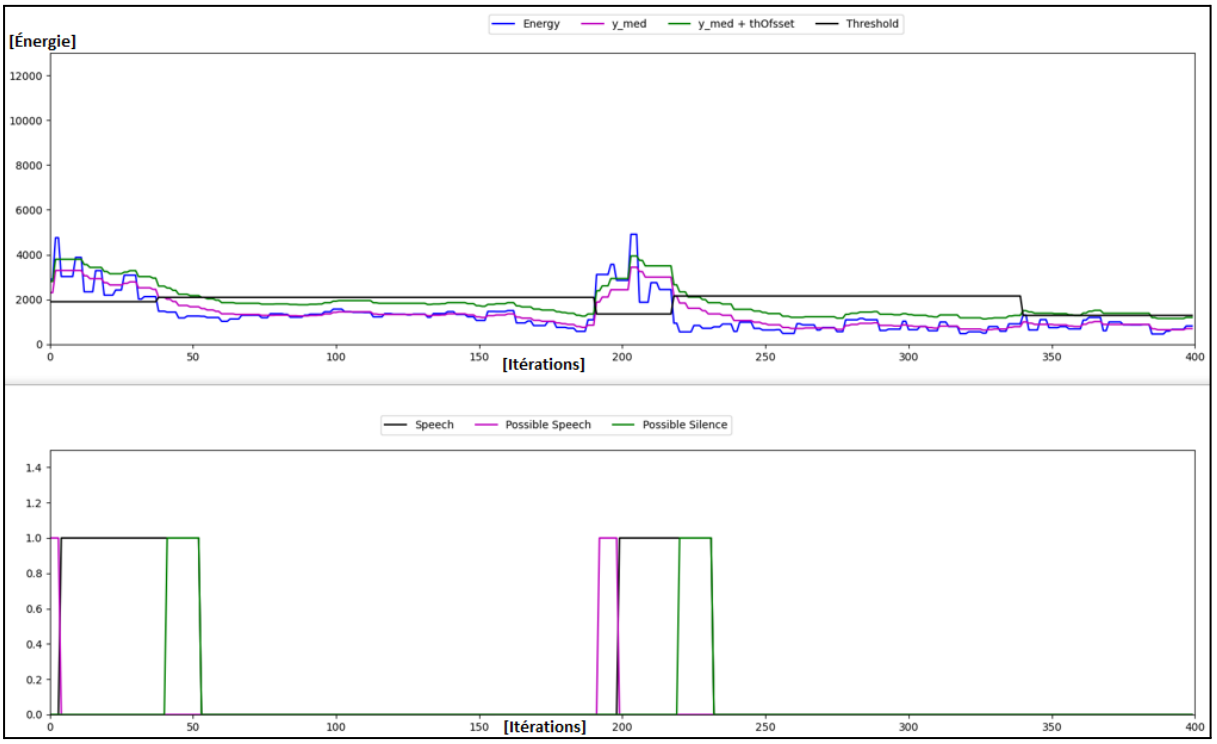
Threshold Offset = 300



Threshold Offset = 400



Threshold Offset = 500



Après avoir effectué les tests précédents pour les cinq valeurs, il a été conclu que pour filtrer la musique de fond, la valeur de la **Threshold Offset** doit être d'au moins 400. Il s'est avéré que les valeurs entre 400 et 500 $[énergie]$ fonctionnent très bien, mais que pour les valeurs supérieures à 500, le locuteur doit parler trop fort pour être écouté.

2.3. Threshold Offset vs Counter Speech

À la fin de la section de l'influence du **Counter Speech**, il a été mentionné qu'il y a un problème dont il faut tenir compte.

Il existe un compromis entre la **Threshold Offset** et le **Counter Speech**. Plus la valeur de **Counter Speech** est élevée, plus le filtrage du bruit impulsif sera efficace, mais aussi plus le locuteur doit parler de manière stable et claire pour être écouté. En effet, si la voix subit la moindre fluctuation (comme lorsque quelqu'un parle en doutant), l'état reviendra au **Silence** et l'enregistrement ne commencera jamais ou échouera.

Une solution peut consister à diminuer la valeur de la **Threshold Offset** mais cela cause un autre inconvénient, le système devient plus susceptible de détecter le bruit de fond (comme la musique ou autre chose) comme un locuteur potentiel et commence l'enregistrement.

De même, la valeur du **Counter Speech** peut être diminuée, mais plus vous la diminuez, plus les bruits impulsifs aléatoires démarreront la reconnaissance. Dans ce scénario, le Threshold Offset doit être augmenté afin de compenser ce fait. Le problème ici est que ce paramètre ne fonctionne pas vraiment bien à cette fin et qu'il est plus probable que la voix du locuteur doive être très forte pour démarrer la reconnaissance.

La tâche la plus difficile ici est donc de trouver une bonne combinaison de ces deux valeurs afin de permettre une bonne détection du son.

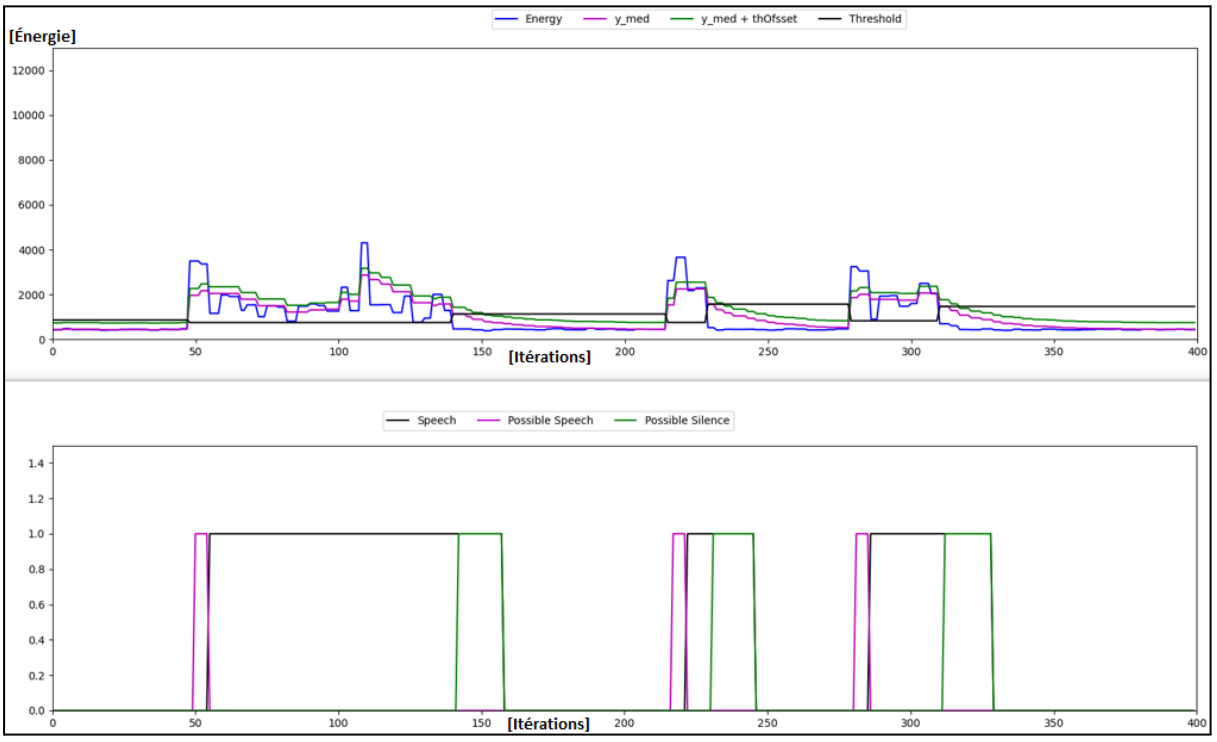
2.4. L'influence du Counter Silence

Enfin, parlons du **Counter Silence**. Ce paramètre peut être testé de manière isolée car le passage du **Silence Possible** à la **Parole** ne dépend que de la valeur de la **Threshold**.

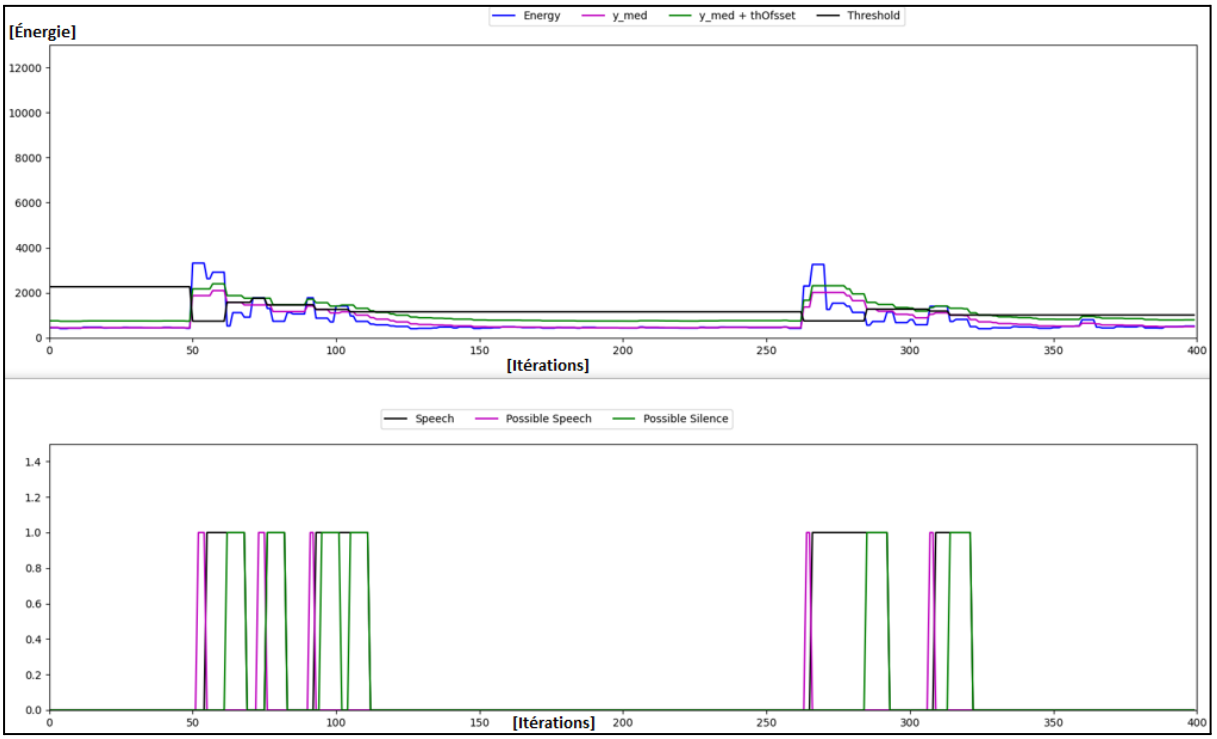
Ainsi, dans cette partie, plusieurs tests ont été effectués en disant de longues phrases tout en vérifiant d'abord que le programme ne change pas à **Silence** alors que le locuteur parlait encore et ensuite en essayant de minimiser le temps qu'il fallait pour arrêter l'enregistrement lorsque le locuteur avait cessé de parler.

Les deux premiers graphiques montrent deux des nombreux tests effectués dans le but de trouver un intervalle où effectuer une recherche plus précise.

Counter Silence = 10



Counter Silence = 5

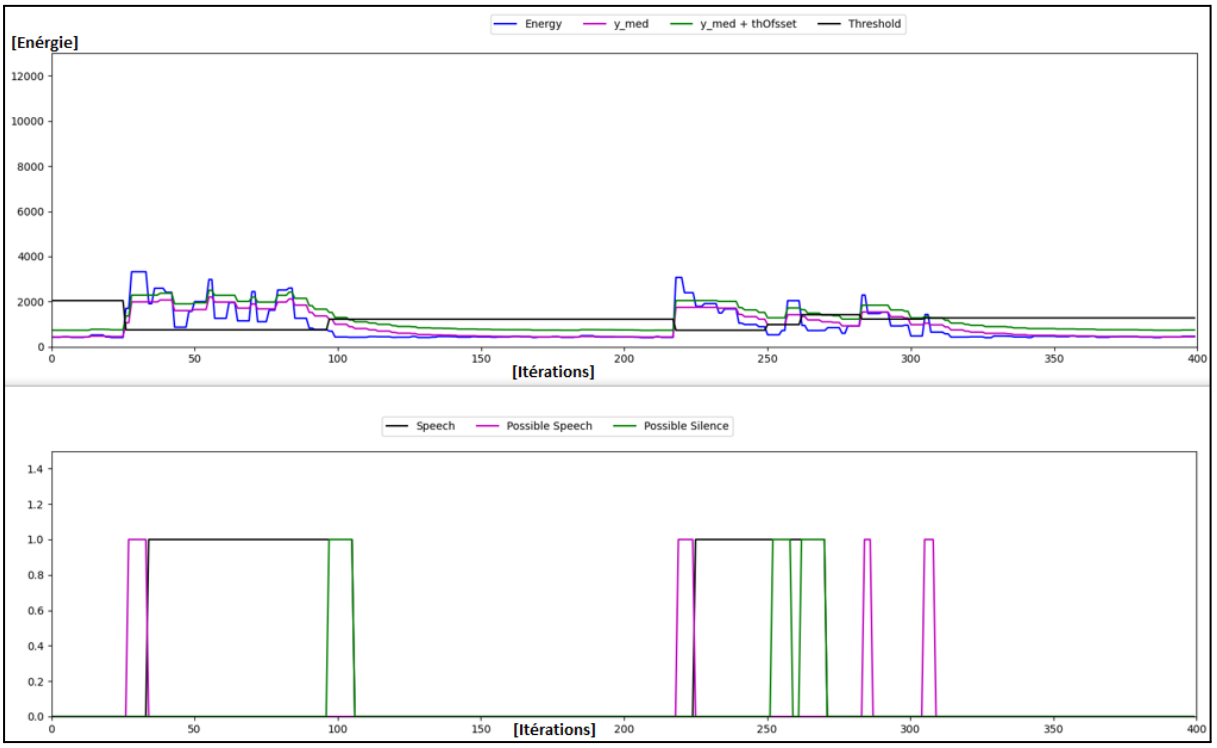


L'intervalle résultant était les valeurs entre 5 et 10 [itérations].

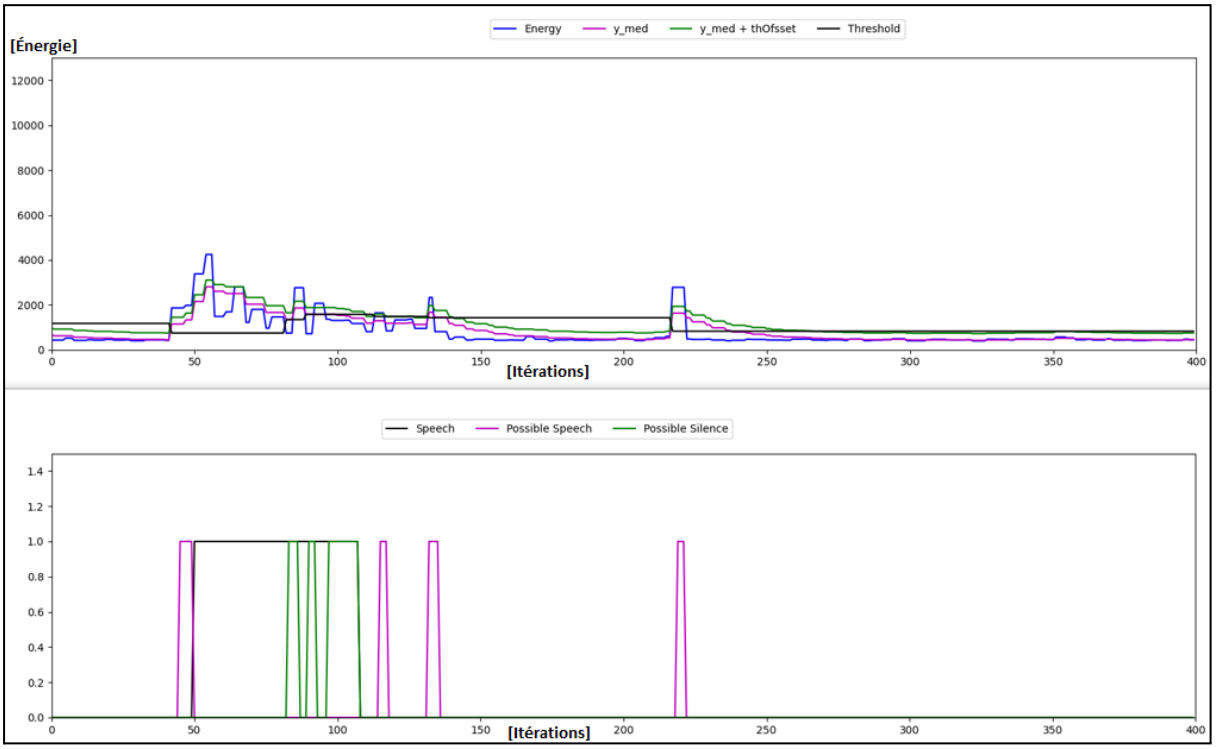
Ensuite, les valeurs 6, 7, 8 et 9 [itérations] ont été testées et il s'est avéré que la valeur 8 était celle qui donnait les meilleurs résultats. Les valeurs inférieures à 8 arrêtaient l'enregistrement trop tôt et les valeurs supérieures mettaient beaucoup de temps à arrêter l'enregistrement.

Voici les graphiques qui montrent les résultats pour les valeurs 6, 7 et 8.

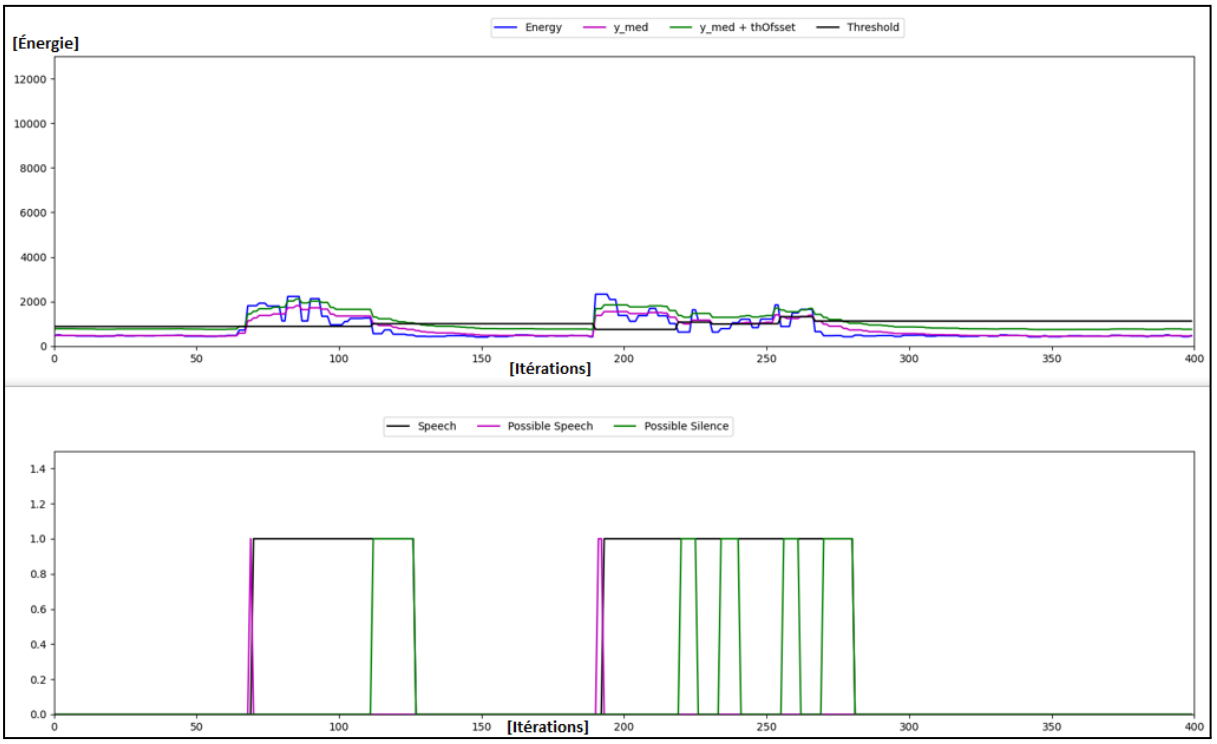
Counter Silence = 6



Counter Silence = 7



Counter Silence = 8



En revenant à l'état de **Silence**, l'enregistrement s'arrête, un fichier wav est créé et la partie du **speech recognition** commence.

3. Reconnaissance Vocale

Après le traitement des sons et l'enregistrement d'un fichier wav, celui-ci doit être traduit en texte. Pour ça, nous avons besoin d'une solution ASR (Reconnaissance vocale automatique). L'ASR est, précisément, un sous-domaine interdisciplinaire de l'informatique et de l' intelligence artificielle qui développe des méthodologies et des technologies qui permettent la reconnaissance et la traduction du langage parlé dans le texte par ordinateur.

Il existe de nombreuses options d'ASR pour cela. Dans ce cas-ci, il devait se conformer à certaines exigences. Tout d'abord, il a dû travailler avec Python 3. Deuxièmement, une solution hors ligne est nécessaire car il est possible que nous n'ayons pas accès à internet lors de la compétition, ou alors que la connexion soit instable. C'était un problème, parce que le meilleur logiciel de reconnaissance vocale, de loin, est de Google, mais il nécessite une connexion Internet stable.

Une autre exigence était que la reconnaissance vocale ne devrait pas être trop lourde pour le CPU, car il n'est pas très puissant et il doit faire d'autres tâches en même temps.

Compte tenu de ces conditions, les softwares de reconnaissance vocale qui ont été sérieusement considérés étaient PocketSphinx, Kaldi, Vosk et Deepspeech. Après quelques tests rapides pour voir si l'un d'eux était meilleur que les autres, il a été décidé d'utiliser Vosk. Il n'y avait pas le temps ou la puissance de calcul pour former un modèle

nous-mêmes, il était donc important qu'ils soient précis hors de la boîte, ce que Vosk était, plus que les autres.

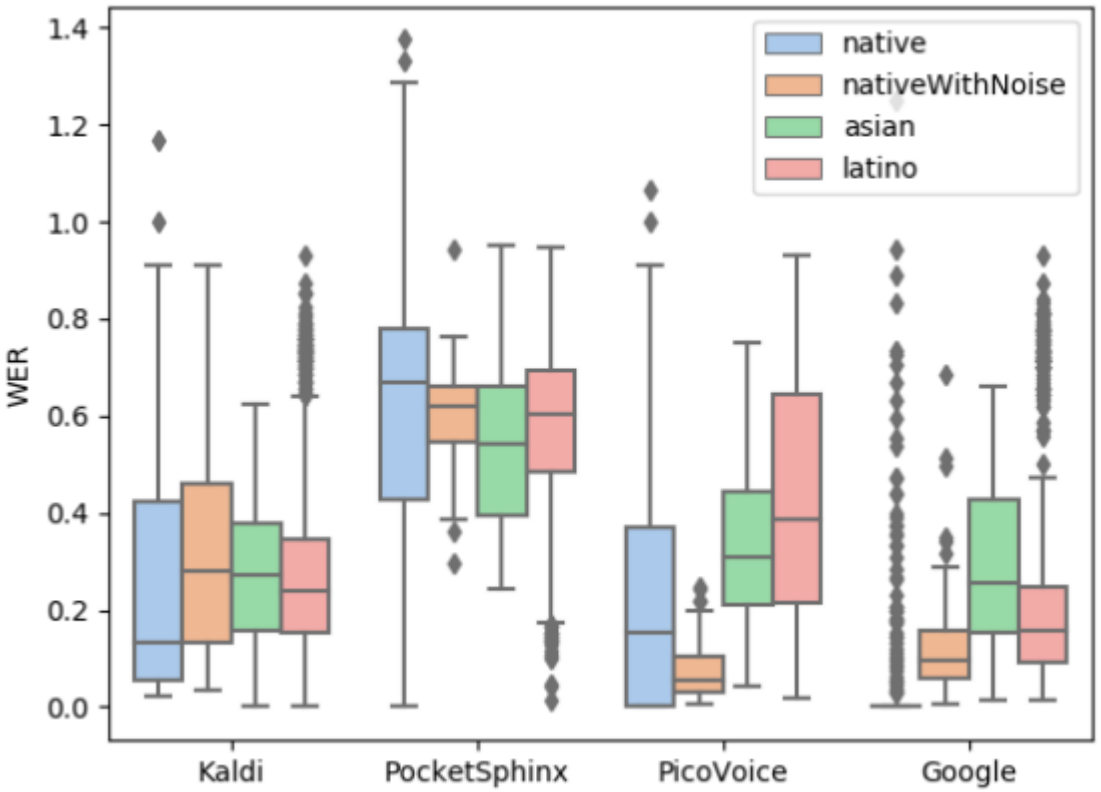
Pour le confirmer, notre solution ASR a été comparée à une référence d'une étude (Brinckhaus et coll., 2021). Ils ont testé Kaldi, Pocketsphinx, Picovoice, et l'ASR de Google, en utilisant la mesure "word error rate", qui est calculé comme :

$$WER = \frac{S + D + I}{N}$$

où,

- S est le nombre de mots substitués,
- D est le nombre de suppressions,
- I est le nombre d'insertions
- N est le nombre de mots dans la phrase.

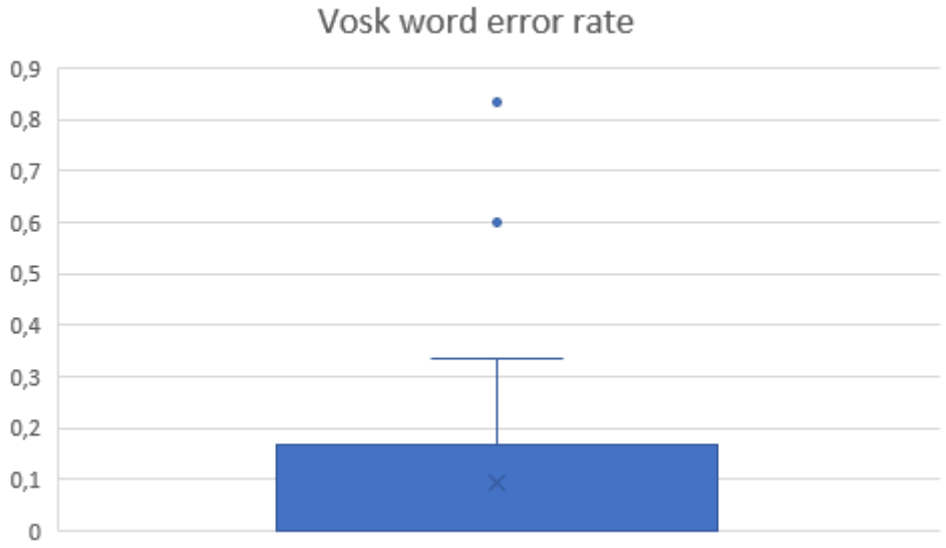
Voici les résultats obtenus :



source: Brinckhaus, Trinidad Barnech, Etcheverry, Andradeet., 2021 - RoboCup@Home: Evaluation of voice recognition systems for domestic service robots and introducing Latino Dataset

Le graphique montre les résultats pour les différents accents de l'anglais. Seul l'américain (native) a été pris en compte dans la comparaison.

Nous avons effectué le même test pour notre solution, en mesurant le WER avec différentes phrases, 101 fois:



Par rapport aux résultats de l'accent natif de l'étude, Vosk a obtenu de meilleurs résultats que les autres logiciels ASR hors ligne. Il est clair que l'ASR de Google est de loin le meilleur, mais comme il a été mentionné, il nécessite une connexion Internet.

Vosk, comme la plupart des systèmes ASR, utilise un modèle formé avec des réseaux neuronaux. Le modèle le plus léger disponible a été utilisé: le modèle "vosk-model-small-en-us-0.15" from *https://alphacephei.com/vosk/models*.

La logique du système est très simple. Le fichier wav créé par le module de traitement du son est ouvert et analysé en petites parties. Les résultats sont transformés en fichier texte. Le code est similaire à l'exemple python du github de Vosk.

4. Traitement Automatique du Langage Naturel (NLP)

C'est la partie où le robot traite tout ce qui a été dit. Un analyseur pour déterminer ce qu'il a été ordonné de faire est la partie principale du NLP. Un code question-réponse a également été élaboré.

4.1. Analyseur syntaxique

Le but de l'analyseur est de déterminer l'intention des phrases prononcées à Pepper. Plus précisément, il doit déterminer quelle action exécuter en fonction de l'ordre donné. La bibliothèque Spacy est utilisée. Spacy utilise un modèle de réseau neuronal pour analyser les phrases. Il classe les mots dans les différentes classes : nom, verbes, adjectifs, etc..., détermine le rôle qu'ils jouent dans une phrase et comment ils sont liés les uns aux autres, entre autres fonctions. Tout comme avec vosk, il existe différents modèles disponibles. Nous avons utilisé le plus léger : "en_core_web_sm", from *https://spacy.io/models/en*.

L’analyseur prendra une chaîne comme entrée et renverra une autre chaîne. Cette chaîne a la forme d’un ou de plusieurs dictionnaires. Par exemple, si la chaîne d’entrée est "take the apple to the table, then follow Valentin", la chaîne retournée sera "{'intent': 'take', 'object': 'apple', 'destination': 'table'}\n{'intent': 'follow', 'person': 'Valentin'}". Les clés du dictionnaire peuvent être : “intent”, “destination”, “object”, “person”, “what”, et “who”. Comme on l’expliquera, tous ne seront pas présents en même temps.

Les intentions possibles ont été classées en six verbes : “go”, “find”, “follow”, “take”, “place”, “say”. “Find” est subdivisé en “find something” et “find someone”, et il en va de même pour “take”. Il y a une longue liste de verbes reconnus mais les actions qui en résultent peuvent être résumées dans ces 6 verbes. Par conséquent, la première chose que le code doit faire est de trouver le verbe (ou les verbes) et de le classer dans l’un de ces 6. S’il y a plus d’un verbe, la phrase est divisée en sous-phrases de sorte que toutes les actions sont enregistrées correctement.

Après avoir déterminé le verbe de base, le reste de l’information est nécessaire. Différentes actions nécessitent des informations différentes, et parfois l’information doit être traitée, ou elle peut être redondante. Les informations nécessaires pour chaque verbe sont :

- “Go”: destination
- “Find something”: objet
- “Find someone”: personne
- “Follow”: personne
- “Take something”: objet, destination (optionnel, mais existera presque toujours)
- “Take someone”: personne, destination
- “Place”: objet, destination
- “Say”: que dire, à qui (optionnel, mais existera presque toujours)

“Go” implicite

Dans certains cas, l’action “go” est implicite, ce qui peut se produire lorsque les options ”take” ou “find” apparaissent. Les phrases du format “take/find (quelque chose/quelqu’un) from (quelque part)”, ou “take/find (quelque chose/quelqu’un) from/on/in/at (quelque part)” seront interprétées comme: premièrement, aller à (quelque part), puis, trouver ou prendre (quelque chose/quelqu’un). Pour ce faire, on cherche les mots “from”, “on”, “in”, “at”. Si l’un d’eux est trouvé, l’action “go” est sauvegardée telle que la destination, puis le reste de la phrase ou de la sous-phrase est analysé.

Destinations

Tous les cas où une destination doit être trouvée sont traités de la même façon. Tout d’abord, le programme vérifie s’il y a un mot qui correspond à une liste prédéterminée de destinations probables. Si cela ne fonctionne pas, il cherchera un adposition et prendra tout ce qui vient après. Les articles sont exclus.

Objets

Lorsqu'un objet doit être déterminé, le programme vérifiera si un objet d'une liste prédéterminée est mentionné. Sinon, il cherchera des mots classés comme objet direct ou objet de préposition et le sauvegardera. Enfin, si l'expression a le mot "it", il va sauvegarder un objet précédemment défini. Par exemple, la phrase "find the cup and give **it** to me" sera divisé en sous-phrases "find the cup" et "give it to me". Dans la première phrase, le "cup" sera l'objet, et dans la seconde, il reconnaîtra le "it" comme l'objet, qui se réfère à l'objet trouvé précédemment ("cup").

Personnes

Quand une personne doit être déterminée, tout comme avec les objets, le programme vérifiera si une liste prédéterminée de noms est mentionnée, et ce sera la personne. Autrement, s'il y a un pronom qui n'est pas "him", "her", ou un pronom possessif, ce sera la personne. Si c'est lui ou elle, le code prendra la personne précédente, comme il l'a fait avec "it" pour les objets.

Action "Say"

Lorsque le verbe dire ou un équivalent apparaît, le programme déterminera toujours quoi dire, et la plupart des fois, à qui le dire. Le "who" sera toujours après le mot "to" ou après le mot "say" ou l'équivalent. Par exemple, "say the time **to him**", ou "**tell him** the time". Par conséquent, il examinera ces positions (après le verbe ou après "to") pour voir s'il trouve une personne, d'une manière semblable à ce qu'il fait pour la clé "person" (il peut s'agir d'un nom, d'un pronom, etc.), et il enregistrera cela comme le "who". Le reste de la phrase est généralement le "what".

4.2. Question-Réponse

Le but de ce module est que le robot peut répondre à différentes questions de culture générale. Afin d'arriver à la bonne réponse le procédure est constitué de trois partie:

- D'une recherche sur **Wikipedia**
- De l'obtention d'une réponse possible grâce à une **modèle BERT**
- De la confirmation de la réponse obtenu ci-dessus avec l'aide d'un **filtre**

4.2.1 Recherche sur **Wikipedia**

D'abord a lieu une classification de la demande selon le type de la question. Les possibles categories sont: "**how**", "**when**", "**who**", "**what**", "**where**", "**which**", "**how many**", "**how much**". Puis une recherche sur Wikipedia est réalisée en utilisant des mots clés dans la question comme des substantifs et adjectifs. Une fois que la recherche prend fin, un sommaire avec tout l'information en rapport à la question est obtenu.

Ensuite toutes les expressions qui sont entre parenthèses sont supprimées du sommaire pour ne pas prendre en compte des clarifications superflues.

Par exemple:

Question: Who was the president of the United States with the longest term in office?

- ➢ Categorie: “**who**”
- ➢ Mots clès: “**the president of the United States with the longest term in office**”

Donc si tu recherche sur Wikipedia les mots clés, le premier résultat qui va apparaitre sera celui-ci:
https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States_by_time_in_office

Puis le code télécharge le sommaire du link ci-dessus et supprime toutes les expressions qui sont entre parenthèses.

4.2.2. Modele BERT

BERT (*Bidirectional Encoder Representations from Transformers*) est une technique basée sur les réseaux neuronaux pour le pré-entraînement du traitement du langage naturel (NLP).

Pour l’obtention de la première réponse possible, le modèle:
"bert-large-cased-whole-word-masking-finetuned-squad" est utilisé. Ce modèle est un modèle question-réponse qui permet d’obtenir une réponse à partir d’une question et d’un contexte où chercher.

L'implémentation et utilisation de ce modèle ci-dessus est très simple en langage de programmation comme python, car il dispose déjà des bibliothèques nécessaires à cet objectif.

Mais le problème ici est que la réponse donnée par le modèle n’est pas toujours la correct donc c’est pour ça qu’il faut d’un filtre de confirmation.

4.2.3. Filtre de confirmation

Le filtre c’est en fait une implémentation très simple. Selon la catégorie obtenue au début, le filtre cherche différents types de mots dans la réponse possible donnée par le modèle.
De la même manière qui a été expliqué dans le Parser, ce ci-dessus est possible grâce à l' utilisation de la bibliothèque Spacy. Les mots suivantes sont “entités” selon la classification de Spacy et ce sont ceux que le code recherche:

- who: PERSON
- where: GPE ou LOC
- when: DATES ou TIMES
- how many, how much: MONEY, QUANTITY ou PERCENT
- what, how, which: PROPN ou NOUN

S' il les trouve, la réponse est correcte.
Par contre, s’il ne trouve rien la réponse est effacé du sommaire et une autre recherche par le modèle BERT est essayée pour arriver à autre possible réponse.

Ce processus, expliqué ci-dessus, est répété jusqu'à ce que la bonne réponse soit trouvée.
Par exemple:

Question: Who is the president of the United States?

Avec le sommaire, trouvé dans la première partie, le modèle BERT peut retourner la réponse “The president of the United States is the head of state and head of government of the United States of America” qui peut être correct mais comme dans la réponse donnée il n’y a pas un entities PERSON la réponse est effacé du sommaire et autre recherche est fait par BERT avec ce sommaire.

De cette façon, nous nous assurons que BERT ne trouve pas la même réponse dans la deuxième boucle. Ce procédé est répété jusqu'à éventuellement il arrive à la bonne réponse qui est “Joe Biden is the 46th and current president of the United States”

5. Conclusion

Dans ce rapport, nous présentons un aperçu du système de dialogue que nous avons développé pour le concours RocoCup@Home. Nous avons fourni une explication générale du fonctionnement de chaque pièce, des outils que nous utilisions et des tests que nous faisions. Nous expliquons d'abord la logique derrière la détection et le traitement de la parole, et les tests que nous avons faits pour calibrer ses paramètres. Ensuite, nous décrivons brièvement la partie discours à texte, et pourquoi nous avons choisi Vosk. Enfin, nous définissons les bases de la solution NLP, à la fois l'analyseur et le code question-réponse que nous avons fait. Les résultats finaux sont satisfaisants, car presque toutes les phrases testées sont analysées correctement, et la solution question-réponse peut répondre à la question la plus élémentaire.

Nous pensons que si, à l'avenir, une autre équipe décide de travailler sur l'NLP, elle sera en mesure de construire à partir de là, car nous avons jeté les bases d'un système de dialogue solide.

6. Références

- *Robocup@Home.* Robocup. Disponible sur : https://athome.robocup.org/ . (Consulté en juin 2022)

- *RoboCup*. Wikipedia. Disponible sur : https://fr.wikipedia.org/wiki/RoboCup . (Consulté en juin 2022)

- *Pepper.* SoftBank Robotics. Disponible sur : https://www.softbankrobotics.com/emea/en/pepper . (Consulté en juin 2022)

- *ROS - Robot Operating System.* Disponible sur : https://www.ros.org/ . (Consulté en mars 2022)

- *Robot Operating System.* Wikipedia. Disponible sur : https://en.wikipedia.org/wiki/Robot_Operating_System . (Consulté en juin 2022)

- *SoftBank Robotics Documentation.* SoftBank Robotics. Disponible sur : http://doc.aldebaran.com/2-5/index_dev_guide.html . (Consulté en mars 2022)

- *Vosk.* Alphacephei. Disponible sur : https://alphacephei.com/vosk/ . (Consulté en mars 2022)

- Brinckhaus E., Barnech G. T., Etcheverry M., et Andrade F.. (2021). *RoboCup@Home: Evaluation of voice recognition systems for domestic service robots and introducing Latino Dataset" .* Disponible sur : https://ieeexplore.ieee.org/document/9605485

- *spaCy 101: Everything you need to know.* Spacy. Disponible sur : https://spacy.io/usage/spacy-101 . (Consulté en mars 2022)

- *BERT large model (cased) whole word masking finetuned on SQuAD.* Hugging Face. Disponible sur : https://huggingface.co/bert-large-cased-whole-word-masking-finetuned-squad (Consulté en avril 2022)