**Université Hassan I**

**Faculté des Sciences et Techniques**

**Département informatique**

Master Réseaux et système informatique

Module : développement mobile

Assistant personnel intelligent en ligne

**Réaliser Par** : LAFRAOUZI Mouhssine

**Encadré Par** : Mr.Y.ELBALOUKI

Table of Contents

[I.](#_heading=h.30j0zll) Concept générale du projet : 3

[II.](#_heading=h.1fob9te) Recherches : 3

[1.](#_heading=h.3znysh7) Recherches globales : comment réaliser un chatbot ? 3

[2.](#_heading=h.2et92p0) Recherches spécifiques : RASA 3

[2.1](#_heading=h.tyjcwt) RASA : 3

[III.](#_heading=h.3dy6vkm) Conception : 3

[1.](#_heading=h.1t3h5sf) Diagramme de cas d’utilisation : 3

[2.](#_heading=h.4d34og8) Diagramme de séquences : 3

1. Concept générale du projet :

[Modification le 28/02/2024]

Le but est de réaliser un chatbot assistant qui réponde aux questions des utilisateurs.

Les questions sont posées soit écrites ou orales.

Pour les types de questions, on verra plus tard.

1. Recherches :
2. Recherches globales : comment réaliser un chatbot ?

[19/02/2024]

Chatbot avec gpt-apis

Lien GitHub contient le code sources :

https://github.com/mnick-yt/Chat-Bot-using-gpt-3.5-turbo/blob/main/ChatGPT%20API/app.py

La vidéo démo :

<https://youtu.be/etTE-mjLiTw?si=3iuhDPA2IyF63DpB>

**Essayer de suivre la démarche avec la même logiciels qu'ils utilisent svp**

**25/02/2024 [modification le 28/02/2024] :**

**Question :** nous avons besoin de quoi pour créer un assistant chatbot ?

**Réponse :**

**Développement à partir de zéro :** Vous pouvez créer un chatbot entièrement à partir de zéro en utilisant un langage de programmation comme Python ou JavaScript, en intégrant des bibliothèques de traitement du langage naturel (NLP) pour comprendre et générer des réponses aux utilisateurs, et en construisant votre propre logique de conversation. Cette approche offre une flexibilité totale mais nécessite généralement plus de temps et de compétences en développement.

**Utilisation de plateformes ou de frameworks de chatbot :** Vous pouvez également utiliser des plateformes de développement de chatbots ou des frameworks comme Dialogflow, Microsoft Bot Framework, Rasa, ou Botpress. Ces outils fournissent des interfaces conviviales, des fonctionnalités avancées pour la gestion des conversations, et souvent des API pour intégrer votre chatbot avec d'autres services et sources de données. Cette approche permet de développer un chatbot plus rapidement en exploitant des fonctionnalités préconstruites, mais elle peut avoir des limitations en termes de personnalisation et de flexibilité.

Voir la conversation : <https://chat.openai.com/share/1f144a41-0866-4161-bcdf-8eadf3e7b115>

1. Recherches spécifiques : RASA

Méthode : Utilisation de la plateforme ou du framework de chatbot "Rasa".

Outils :

* Front-end : HTML, CSS, JavaScript.
* Backend : Python.
* Framework : Rasa.

[ 28/02/2024 ]:

* 1. RASA :

<https://rasa.com/docs/>

**Fonctionnement de RASA**

Rasa est un outil de création de chatbots open source, nous allons expliquer

son fonctionnement de façon plus précise mais il ne nous sera pas possible d’at-

teindre l’exhaustivité de la documentation officielle dont nous nous inspirons .

**Architecture générale**

Rasa (Bocklisch et al., 2017) est un logiciel open source qui propose deux

modules : Rasa NLU et Rasa Core. Le premier permet de mettre en place la

compréhension du langage naturel tandis que le second s’occupe du « dialog

management », la gestion des dialogues. L’objectif, selon les créateurs, est de

combler le fossé entre la recherche et l’application et d’amener les récentes avan-

cées du machine learning à des personnes inexpérimentées qui souhaitent im-

plémenter des agents conversationnels. Les modules NLU et Core peuvent être

utilisés ensemble ou séparément, la modularité étant un parti pris pour donner

un maximum de liberté au développeur. Il est donc possible d’utiliser le module

Core avec un autre système NLU et inversement. Le code est entièrement écrit

en Python et il est possible de le modifier aisément.

Un agent conversationnel procède généralement de la même manière pour

mener à bien la tâche qui lui est confiée. Le système que nous utiliserons, Rasa,

procède comme suit :

— Réception du message de l’utilisateur, classification de l’intent et extrac-

tion des entités (NLU) ;

— Un tracker maintient la discussion et est notifié du message reçu ;

— Le « policy » reçoit l’état courant du tracker et choisit une action à

prendre ;

— L’action est ensuite enregistrée dans le tracker et exécutée (la fonction

peut générer l’envoi d’une réponse, un calcul ou autre chose).

**Rasa NLU**

Les phrases données en entrée au chatbot sont représentées par une mise

en commun de vecteurs de mots pour chaque token. Pour ce faire, Rasa utilise

(dans son modèle recommandé) des words embeddings préentrainés (GloVe). Se-

lon eux, le classificateur d’intents est robuste. Les intents et autres entités sont

alors extraits et donnés au module Core qui s’occupe de décider de la prochaine

action selon ce qu’il a reçu. Rasa NLU peut être utilisé avec plusieurs modèles

(ou pipelines 2 : tensorflow\_embedding ou bien spacy\_sklearn).

1. https ://rasa.com/

2. https ://rasa.com/docs/nlu/choosing\_pipeline/

**FONCTIONNEMENT DE RASA**

leurs propres avantages et spécificités : spacy\_sklearn utilise des words embed-

dings déjà préentrainés (GloVe) alors que TensorFlow les construit directement

depuis les données du chatbot. Spacy\_sklearn devrait donc être préféré lorsque

la base de données est assez faible alors que TensorFlow peut être utilisé avec

beaucoup d’intérêt si la base de données disponible est suffisamment conséquente

et pertinente. Nous allons nous intéresser plus spécifiquement à spacy\_sklearn,

le pipeline recommandé :

* Tokenisation, POS tagging et vectorisation de chaque token grâce à GloVe

pour créer une représentation de la phrase ;

* Scikit-learn classifier entraine ensuite un estimateur pour le jeu de don-nées. Par défaut, il consiste en un multiclass support vector (type par-

ticulier de SVM) dans le cadre de la classification d’intents. Les entités sont reconnues grâce au module ner\_crf qui a pour tâche d’entrainer un conditional random fiel (CRF) qui utilise les tokens et les POS tags.D’autres composants permettent de gérer les out-of-vocabulary (OOV) et proposent de nombreuses personnalisations.

De façon générale, Rasa encourage les développeurs à entrainer leurs chatbots de

façon interactive, en corrigeant ses erreurs. Cette méthode (machine teaching)

permet d’améliorer significativement les résultats obtenus en « enseignant » les

bonnes réponses à son chatbot. Rasa Core, pour sa part, ne reçoit que les intents

et les entités reconnues.

Rasa Core

Ce module retiendra moins notre attention car il n’a pas d’influence sur la

classification des intents. En effet, il intervient en aval.

La question du dialogue management est considérée comme un problème de

classification. Pour chaque itération, Rasa Core prédit quelle action doit être

exécutée depuis une liste. Il utilise pour ce faire les intents et les entités du

message le plus récent. L’état du dialogue est conservé dans un tracker object,

il en existe un seul par discussion.

Les données cruciales qu’utilise le Core sont les stories. Une story est un en-

chainement logique d’intents permettant de construire un fil de dialogue. Elles

sont particulièrement puissantes dans les systèmes goal oriented car elles parti-

cipent grandement à la bonne gestion du dialogue.

## conso path 1 -> nom de la story

\* greet -> nom de l’intent attendu (bonjour)

- utter\_greet -> action à exécuter (accueillir l’utilisateur)

\* conso

- utter\_conso

- utter\_goodbye

Cette story prend donc en charge une discussion de ce type :

Utilisateur : - Bonjour Bot!

Chatbot : - Bonjour, comment puis-je vous aider?

U : - Je voudrais connaitre la consommation de cette voiture.

C : - Cette voiture consomme 4,7 litres pour cent kilomètres.

C : - Au revoir!

La classification des intents intervient donc en amont et donne au Core l’intent

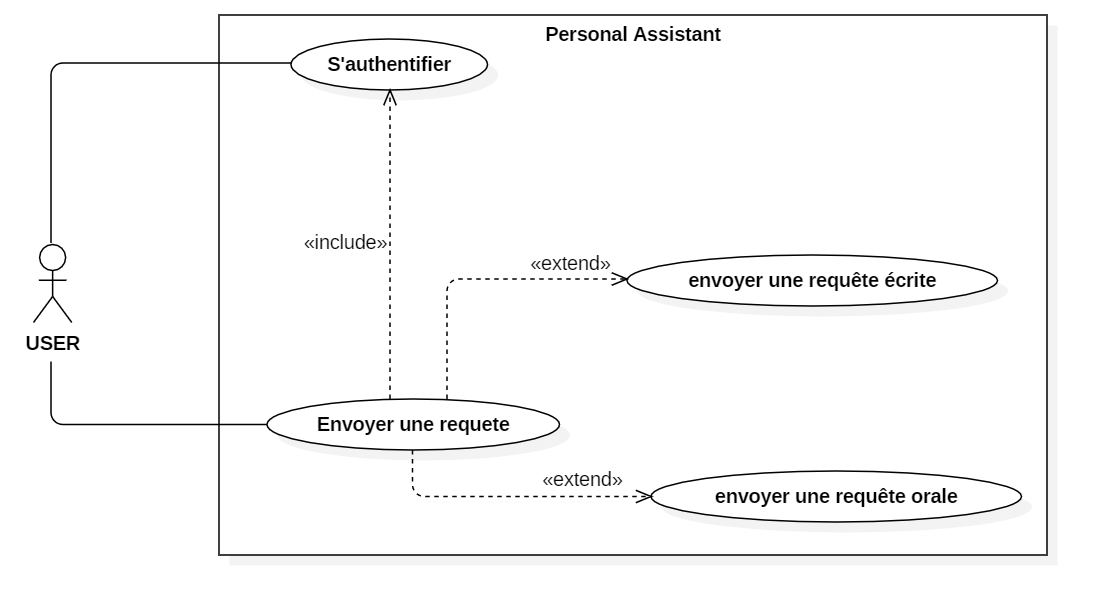
reconnu afin de décider de l’action suivante la plus probable.

recherche youtube : <https://www.youtube.com/results?search_query=Chatbots+bas%C3%A9s+sur+des+mod%C3%A8les+pr%C3%A9-entrain%C3%A9s>

<https://www.youtube.com/watch?v=GwaSJUlB8oA&list=PLtFHvora00y8NBwCMoNnPqii-y2-gyl5p&index=2>

**20/02/2024 [modification le 25/02/2024]**

1. Conception :
2. Diagramme de cas d’utilisation :

****

1. Diagramme de séquences :

[ 27/02/2024 ]: