

## Table des matières

|  |    |
|--|----|
| <b>Introduction générale :</b>   | 4  |
| <b>Chapitre 1</b>  | 6  |
| <b>1 Étude préliminaire</b>  | 6  |
| <b>1.1 Introduction</b>  | 6  |
| <b>1.2 Présentation de la société</b>  | 6  |
| <b>1.3 Présentation du projet :</b>  | 7  |
| <b>1.4 L'état de l'art :</b>   | 7  |
| <b>1.4.1 L'analyse de l'existant :</b>   | 7  |
| <b>1.4.2 Solutions proposées :</b>   | 8  |
| <b>1.4.3 Travail à faire :</b>   | 8  |
| <b>1.5 Analyse et spécification des besoins :</b>                                    | 8  |
| <b>1.5.1 Identification des acteurs :</b>  | 8  |
| <b>1.5.2 Exigences fonctionnelles :</b>  | 8  |
| <b>1.5.3 Exigences non fonctionnelles :</b>  | 9  |
| <b>1.5.4 Diagramme de cas d'utilisation général</b>                                  | 9  |
| <b>1.6 Conclusion :</b>  | 9  |
| <b>Chapitre 2</b>  | 10 |
| <b>2 Assistants personnels et Rasa Framework</b>                                     | 10 |
| <b>2.1 Introduction</b>  | 10 |
| <b>2.2 Présentation :</b>  | 10 |
| <b>2.3 Définition</b>  | 11 |
| <b>2.4 Histoire du chatbot et avènement de l'intelligence artificielle</b>           | 11 |
| <b>2.5 Exemples de chatbots :</b>  | 13 |
| <b>2.6 Identifier les opportunités pour un chatbot d'Intelligence Artificielle :</b> | 13 |
| <b>2.7 Développement de chatbots</b>   | 13 |
| <b>2.7.1 Types de chatbots</b>   | 14 |
| <b>2.7.2 Principales applications des chatbots :</b>                                 | 14 |
| <b>2.8 L'architecture des chatbots :</b>   | 15 |
| <b>2.9 Chatbots intelligents</b>   | 16 |
| <b>2.9.1 Chatbots performants</b>  | 16 |
| <b>2.9.2 Chatbots : à quel point sont-ils intelligents ?</b>                         | 16 |
| <b>2.9.3 Les robots du futur</b>   | 17 |
| <b>2.10 Chatbots pour les webinaires</b>   | 17 |
| <b>2.10.1 Webinaires</b>   | 17 |

|                 |   |    |
|-----------------|---|----|
| 2.10.2          | Logiciel/site Web du webinaire .....                            | 17 |
| 2.10.3          | Fonctionnalité de webinaire.....                                | 17 |
| 2.10.4          | Avantages et inconvénients des webinaires :.....                | 18 |
| 2.10.5          | Webinaires et chatbots :.....                                   | 18 |
| 2.11            | Cadre Rasa.....   | 20 |
| 2.11.1          | Définition.....   | 20 |
| 2.11.2          | Explorer Rasa .....   | 20 |
| 2.11.3          | Structure du projet Rasa .....                                  | 21 |
| 2.12            | Chatbots à commande vocale .....                                | 23 |
| 2.12.1          | Parole en texte.....  | 23 |
| 2.12.2          | Synthèse vocale .....   | 23 |
| 2.12.3          | Structure d'une conversation vocale.....                        | 24 |
| 2.13            | Conclusion.....   | 24 |
| Chapitre 3..... |   | 25 |
| 3               | Réalisation Partie 1 : Créer un Chatbot avec Rasa.....          | 25 |
| 3.1             | Introduction .....  | 25 |
| 3.2             | Environnement de travail .....                                  | 25 |
| 3.2.1           | Environnement de développement.....                             | 25 |
| 3.2.2           | Environnement en ligne .....                                    | 26 |
| 3.3             | Modèle NLU.....   | 27 |
| 3.3.1           | Compréhension du langage naturel .....                          | 28 |
| 3.3.2           | Génération des données de formation NLU pour notre chatbot..... | 28 |
| 3.3.3           | Pipeline de formation NLU.....                                  | 29 |
| 3.3.4           | Choix d'une configuration de pipeline : .....                   | 29 |
| 3.3.5           | Composants du pipeline de formation .....                       | 30 |
| 3.3.6           | Test du modèle .....  | 31 |
| 3.4             | Gestion des dialogues .....                                     | 31 |
| 3.4.1           | Stories :.....  | 31 |
| 3.4.2           | Domaine, actions personnalisées et emplacements(slots).....     | 32 |
| 3.4.3           | Politiques de dialogue.....                                     | 35 |
| 3.4.4           | Échouer gracieusement à Rasa.....                               | 36 |
| 3.4.5           | Former et tester l'assistant .....                              | 36 |
| 3.5             | Base de données .....   | 37 |
| 3.5.1           | MongoDB .....   | 37 |
| 3.5.2           | Traitement des données .....                                    | 37 |
| 3.6             | Conclusion.....   | 38 |

|   |           |
|---|-----------|
| <b>Chapitre 4.....</b>  | <b>39</b> |
| <b>4    Realisation de Partie 2 : Implémentation and Déploiement.....</b> | <b>39</b> |
| <b>4.1    Introduction :.....</b>   | <b>39</b> |
| <b>4.2    Rasa X.....</b>   | <b>39</b> |
| <b>4.2.1    Obtenir de bonnes données d'entraînement : .....</b>          | <b>39</b> |
| <b>4.2.2    Déploiement de Rasa X.....</b>                                | <b>39</b> |
| <b>4.3    Connexion aux canaux de messagerie.....</b>                     | <b>45</b> |
| <b>4.3.1    Meilleur logiciel de webinaire sur le marché .....</b>        | <b>45</b> |
| <b>4.4    Conclusion.....</b>   | <b>46</b> |
| <b>Conclusion générale : .....</b>  | <b>47</b> |
| <b>Références .....</b>   | <b>48</b> |
| <b>Résumé .....</b>   | <b>49</b> |

# Liste des figures

|   |    |
|---|----|
| Figure 1.1 : logo d'acarobots Technologie.....            | 6  |
| Figure 2.1 : Digital Einstein.....                        | 10 |
| Figure 2.2 : histoire de chatbot .....                    | 12 |
| Figure 2.3 : architecture de chatbot.....                 | 15 |
| Figure 2.4 : logo de rasa .....                           | 20 |
| Figure 2.5 : les étapes d'une conversation vocale .....   | 24 |
| Figure 3.1 : les caractéristiques de pc.....              | 25 |
| Figure 3.2 : environnement logiciel.....                  | 25 |
| Figure 3.3 : génération de données de formation NLU ..... | 28 |
| Figure 3.4 : configuration de pipeline .....              | 29 |
| Figure 3.5 : tester le modèle.....                        | 31 |
| Figure 3.6 : Rasa core : Stories .....                    | 32 |
| Figure 3.7 : Rasa core : Domain File .....                | 33 |
| Figure 3.8 : les réponses de chatbot .....                | 33 |
| Figure 3.9 : Rasa core : fichier des actions .....        | 34 |
| Figure 3.10 : Rasa core : Policies .....                  | 35 |
| Figure 3.11 : test .....                                  | 37 |
| Figure 4.1 : créer une machine virtuelle .....            | 40 |
| Figure 4.2 : connexion SSH .....                          | 40 |
| Figure 4.3 : Rasa X.....                                  | 41 |
| Figure 4.4 : GitHub .....                                 | 42 |
| Figure 4.5 : Rasa X.....                                  | 42 |
| Figure 4.6 : fichier docker .....                         | 43 |
| Figure 4.7 : parler avec le chatbot.....                  | 43 |
| Figure 4.8 : tester la conversation.....                  | 44 |
| Figure 4.9 : canaux de messagerie .....                   | 45 |

## **Introduction générale :**

Les chatbots, ces agents conversationnels intelligents, étaient initialement utilisés pour le service après-vente. Les bots permettent désormais de proposer une expérience beaucoup plus riche aux internautes. Ils peuvent ainsi devenir un formidable levier marketing pour conseiller vos clients, les informer ou effectuer des réservations depuis un site, une application mobile ou de messagerie, prendre compte du contexte et des préférences de l'utilisateur.

Les chatbots ne sont pas un nouveau développement, mais plutôt un simulateur qui peut comprendre le langage humain, le traiter et interagir avec les gens lors de certaines tâches. Par exemple, les chatbots peuvent être utilisés pour diriger des équipes de soutien. Le premier chatbot a été créé par Joseph Wiesenbaum en 1966 et s'appelait Eliza. Tout a commencé quand Alan Turing a publié un article "Computer Machinery and Intelligence" et a posé une question intéressante : "Can machine Think ?", depuis lors, certains chatbots ont surpassé leurs prédécesseurs en les rendant plus faciles à comprendre et une technologie plus avancée. Ces avancées nous ont permis d'adopter une manière de communication avec les chatbots, qui devient aussi normale et naturelle que les autres.

Aujourd'hui, presque toutes les entreprises ont des chatbots qui peuvent attirer des utilisateurs et fournir des services clients en répondant à leurs questions. Selon un rapport Gartner, d'ici 2022, les chatbots traiteront 85 % des appels d'assistance. Nous avons des chatbots presque partout. Mais cela ne signifie pas nécessairement que tout ira bien. Le défi ici n'est pas de développer un chatbot, mais un bot qui fonctionne bien.

Dans ce cadre, mon projet de fin d'études intitulé " Assistants Personnels et Chatbots utilisant RASA Framework dans un webinaire d'accès à distance ", réalisé à la société acarobotics, vise à développer un chatbot basé sur le framework rasa et à appliquer l'intelligence artificielle et l'apprentissage en profondeur pour obtenir une performance ou un bon fonctionnement d'assistant. Plus précisément, il s'agit d'un agent conversationnel intelligent (chatbot) qui utilise les nouvelles technologies associées au naturel langage processing (NLP), naturel langage understanding (NLU), Machine Learning et Deep Learning, pour apporter des réponses précises et aider les participants au webinaire à apprendre et à progresser leurs connaissances dans n'importe quel domaine.

Le rapport est divisé en quatre chapitres. Le premier se concentre sur le contexte de projet dans lequel nous décrivons la description globale du projet, les solutions existantes et la solution proposée. Puis, le deuxième, présente les chatbots en détails : Concepts de chatbot, différents types d'assistants, quelques cas d'utilisation pour les bots et leur intelligence. Ensuite, comprendre le framework Rasa.

Le troisième détaille la première étape de la mise en œuvre ou bien l'implémentation, qui consiste à élaborer un texte et un chatbot conversationnel vocal utilisant le framework Rasa.

La quatrième traite du développement, du test et du déploiement du chatbot. En plus de présenter les différentes technologies mises en œuvre et de détailler les outils matériels et logiciels qui ont servi à la réalisation de ce projet.

Enfin, ce rapport se termine par une conclusion générale et les perspectives proposées.

# Chapitre 1

## 1 Étude préliminaire

### 1.1 Introduction

L'objectif de ce chapitre est de présenter le projet, nous commençons par une idée générale sur le contexte du projet, puis nous présentons l'organisme d'accueil et enfin nous donnons une description détaillée des travaux à effectuer.

### 1.2 Présentation de la société

La société Acarobotics Technologies est spécialisée dans la robotique académique et industrielle. Elle s'intéresse également à l'automatisme industriel.

Acarobotics touche tous les domaines technologiques de la robotique académique à la robotique industrielle. Elle a un savoir-faire important concernant l'électronique. Elle dispose d'une gamme de cartes et de kits robotiques qui répondent à toutes les demandes clients.

C'est la première société tunisienne spécialisée dans l'Education 4.0. Conception et réalisation des programmes, outils et plateforme pour la Robotique, développement informatique, intelligence artificielle et conception des jeux vidéo et présente des curriculums en robotique "Bots for All by AcaROBOTICS" avec une gamme de kit robotique éducatif "AcaKIDS" et un curriculum en informatique "Future Kids by AcaROBOTICS" qui présente C2i, codage, infographie et l'enfant entrepreneur"



Figure 1.1 : logo d'acarobotics Technologie

### Fiche d'identité AcaRobotics :

| Catégorie       | Etablissement privé   |
|-----------------|---|
| <u>Adresse</u>  | 6 Rue Jean Jacques Rousseau, Montplaisir Espace Tunis, Bloc D, Monplaisir, Tunis 1073 |
| Gouvernera      | TUNIS   |
| Site web        | <a href="http://www.aca-robotics.com/">http://www.aca-robotics.com/</a>               |
| Gmail           | <a href="mailto:acaroboticstechnology@gmail.com">acaroboticstechnology@gmail.com</a>  |
| Forme juridique | (S.A.R.L) société à responsabilité limité   |

|          |   |
|----------|---|
| LinkedIn | <a href="https://www.linkedin.com/company/acarobotics-technologies/">https://www.linkedin.com/company/acarobotics-technologies/</a> |
|----------|---|

### **1.3 Présentation du projet :**

Les clients apprécient les chatbots car ils sont rapides, intuitifs et pratiques. Pour les entreprises, les chatbots IA leur permettent de créer une expérience client plus personnalisée et attrayante, ce qui, à son tour, offre une mine d'informations précieuses sur les consommateurs, ce qui est essentiel pour développer leur entreprise.

L'avènement des smartphones, des appareils portables et de l'Internet des objets (IoT) a changé le paysage technologique de façon spectaculaire ces dernières années, alors que les artefacts numériques sont devenus plus petits et leur puissance de calcul a augmenté. Cependant, les applications mobiles et les processus gourmands en données ne sont pas parallèles. Naviguer les menus complexes n'est pas l'expérience utilisateur rapide et fluide dont les entreprises d'aujourd'hui ont besoin. En outre, les consommateurs ne se contentent plus d'être contraints par les moyens de communication choisis par une organisation. Ils veulent s'interfacer avec la technologie à travers un grand nombre de canaux. Un robot IA conversationnel offre un moyen de résolution de ces problèmes en permettant aux clients de demander simplement tout ce dont ils ont besoin, sur plusieurs canaux, où qu'ils se trouvent, le jour comme la nuit.

### **1.4 L'état de l'art :**

#### **1.4.1 L'analyse de l'existant :**

##### **Assistante clientèle :**

Les clients sont assistés par des assistants clients en traitant les plaintes, en répondant aux questions et percevoir les paiements sur les comptes en difficulté en personne et par téléphone. Centres d'appels, agences de recouvrement, compagnies d'assurance, centres financiers, magasins de détail et entreprises qui rendent service un volume élevé de clients embauchent des assistants clients pour un travail à temps plein et à temps partiel. Les assistants clients travaillent généralement pendant les heures ouvrables de jour, bien que cela varie en fonction de l'entreprise contractante.

##### **Les défis du service client :**

À mesure que la concurrence augmente, un bon service à la clientèle est devenu la pierre angulaire du succès d'une entreprise. Mais fournir des produits de qualité associés à un service client exceptionnel est un grand défi. Ayons un coup d'œil aux principaux défis du service client que vous pourriez rencontrer

1. Ne pas connaître la réponse à une question : Il y aura des moments où les clients surprendre l'assistant client avec des questions auxquelles il ne peut pas répondre.
2. Répondre aux attentes des clients : Parfois, les clients ont du mal à expliquer leur problème. C'est tout à fait normal. Ils ne connaissent peut-être pas vos processus et termes techniques, seulement qu'ils sont déçus que votre produit ou service n'aient pas répondu à leurs attentes.
3. Parler aux clients en colère : les responsables du service client qualifiés sont primordiales pour traiter les clients en colère. Ces clients ont besoin de quelqu'un pour

répertorier le service client défis et leur offrir une oreille empathique. Ils souffrent et ce n'est pas de leur faute.

#### **1.4.2 Solutions proposées :**

Compte tenu de la valeur des webinaires, il est important de les mener en douceur.

Effectivement, les assistants virtuels sont relativement peu coûteux pour fournir la source et l'embauche des experts dans tous les aspects de la gestion d'un webinar peuvent être une ressource inestimable. Nous n'avons pas à prendre en charge le webinar uniquement par le présentateur, car cela entraîne souvent des problèmes imprévus cela peut conduire à la frustration du public et peut entraver les objectifs du webinar. Le principal moyen pour les chatbots d'aider est d'arrêter les demandes retardées des clients, qui augmente le coût des appels d'assistance à un niveau ingérable. Les chatbots ne peuvent pas tout faire, mais ils peuvent réduire la pression sur les dépenses des agents en direct. Ils peuvent faire beaucoup plus, y compris répondre à toute question sur l'entreprise et ses webinaires, travailler rapidement, offrant un accès 24 heures sur 24, 7 jours sur 7, ce qui permet d'évoluer plus facilement.

#### **1.4.3 Travail à faire :**

La tâche principale est de créer un assistant virtuel, l'étape suivante consiste à ajouter l'interface chatbot à l'application dans laquelle le chatbot sera implémenté, dont la plupart est le chatbot lui-même. La principale préoccupation consiste à créer et à développer un chatbot efficace en utilisant le framework Rasa. L'assistant sera testé pour tirer parti de la façon dont il interagit avec de vrais testeurs. Ce processus aidera à améliorer les performances de l'assistant. À la fin du projet nous attendons une phase de mise en œuvre du chatbot se connectant à de nombreuses plateformes.

### **1.5 Analyse et spécification des besoins :**

Cette partie est dédiée à l'analyse et à la spécification des besoins de l'application. Commencant par présenter les exigences fonctionnelles et non fonctionnelles de l'application. Ensuite l'architecture et le mode de fonctionnement à l'aide du diagramme de cas d'utilisation.

#### **1.5.1 Identification des acteurs :**

- ***Participants au webinar*** : les principaux utilisateurs du chatbot. Ils interagiront avec lui pendant que le présentateur du webinar parle.
- ***L'animateur du webinar*** : est une personne qui sera sur l'écran du site (en direct) une fois qu'un webinar commence.

#### **1.5.2 Exigences fonctionnelles :**

Les besoins fonctionnels expriment l'action à effectuer par le système en répondant à une demande, ce sont exactement les sorties qui sont produites pour un ensemble donné d'entrées. Cette application doit couvrir principalement les besoins fonctionnels suivants :

- ***Interface de streaming*** : cette interface sera l'interface principale de la plateforme pour que les participants puissent le suivre et le voir lors d'un webinar, le présentateur doit ouvrir le microphone et la caméra au début de la session.
- ***Inviter des participants*** : le présentateur du webinar doit inviter ses participants, il doit leur envoyer par e-mail un e-mail contenant le lien du webinar.
- ***Interface d'assistant personnel pour le webinar*** : avant le début du webinar les participants peuvent poser des questions sur le webinar lui-même : quel est le prochain



webinar ? qui est le présentateur ? quand le webinaire commence, et ainsi de suite... dans quel cas l'assistant doit répondre à vos questions.

Par la suite et après le début du webinaire, le même chatbot doit prendre en charge le webinaire avec le présentateur et lui aider à répondre aux questions des participants.

- **Chatbot à commande vocale** : les chatbots à commande vocale sont ceux qui peuvent interagir et communiquer par la voix. Ils sont capables d'accepter la commande dans une forme orale ou écrite. Ils sont programmés pour répondre vocalement. Cela peut être en outre classés en deux types : un premier qui répond via le texte et la voix ensemble. Le deuxième, répond uniquement par la voix. Dans ce projet, nous utiliserons le premier (texte et voix).

### 1.5.3 Exigences non fonctionnelles :

Après avoir établi les exigences fonctionnelles du projet, nous procédons à la description des exigences non fonctionnelles qui concernent les aspects comportementaux de l'application dans Problèmes de performances. Ces exigences peuvent être le résultat de restrictions de mise en œuvre (langage de programmation, base de données...), qui reposent principalement sur ces points :

- **Ergonomique** : Le système doit comporter des interfaces conviviales et ergonomiques qui sont faciles à utiliser.
- **Performance** : Les transmissions et récupérations de données doivent être réalisées dans un délai raisonnable.
- **SimPLICITÉ** : La solution doit être la plus simple possible et doit être facile à maintenir et à déboguer grâce à la compréhensibilité et la clarté de son code et de sa conception.
- **Extensibilité** : Le système doit pouvoir subir d'éventuelles extensions, réutilisables et capable de s'adapter aux changements constants

### 1.5.4 Diagramme de cas d'utilisation général

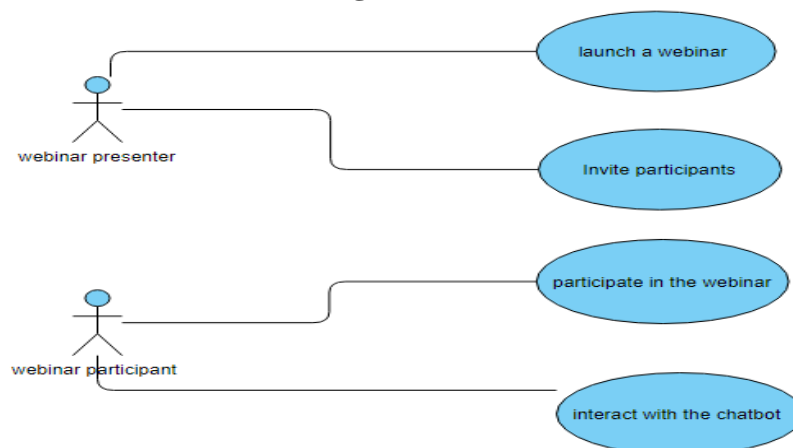


Figure 1.2 : Diagramme de cas d'utilisation

La figure (1.2) représente le schéma général des cas d'utilisation qui résume les principales fonctionnalités offertes par notre plateforme/chatbot et les principaux acteurs.

### 1.6 Conclusion :

Dans ce chapitre, nous avons donné une idée générale du projet au sein de l'entreprise acarobotics technologies en précisant l'analyse et la critique de l'existant ainsi que le concept

général de notre projet. Enfin nous présentons le cahier des charges et le diagramme de cas d'utilisation global.

## Chapitre 2

### 2 Assistants personnels et Rasa Framework

#### 2.1 Introduction

Dans ce chapitre, nous allons présenter les chatbots en détail : Concepts de chatbot, différents types des assistants, quelques cas d'utilisation pour les bots, à quel point sont-ils intelligents, chatbot pour les webinaires et par la suite nous présenterons le framework rasa en détails.

#### 2.2 Présentation :

UneeQ, une entreprise humaine numérique fondée aux États-Unis et en New-Zélande, annonce aujourd'hui le lancement de Digital Einstein.

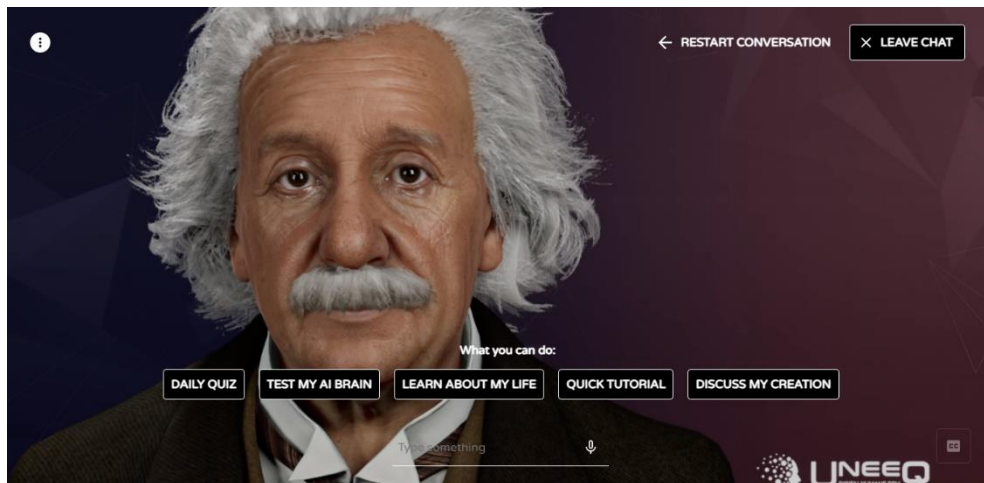


Figure 2.1 : Digital Einstein

À l'occasion du 100e anniversaire du prix Nobel de physique d'Albert Einstein, UneeQ a remasterisé Albert Einstein en tant qu'être humain numérique guidé par l'intelligence artificielle (IA) conversationnelle et expérientielle pour créer une expérience significative et sans précédent avec l'un des plus grands esprits de l'histoire. En tant qu'exemple d'IA expérientielle, Digital Einstein représente la prochaine tendance de développement de l'interaction homme-ordinateur. Il utilise des humains numériques pour générer des interactions centrées sur la personnalité, en faisant participer les clients, les patients, les étudiants et les utilisateurs finaux à la transaction interaction émotionnelle significative. (1)

Dans des analyses plus approfondies, Digital Einstein est un Chatbot et plus précisément un chatbot vocal qui peut interagir avec les gens. Il a été créé à l'aide de nombreux concepts d'IA tels que l'apprentissage de machine (machine Learning) (ML), le traitement du langage naturel (Natural Language Processing) (NLP), la synthèse vocale (text-to speech) (TTS), la synthèse vocale (speech-to text) (STT), etc....

## 2.3 Définition

Les chatbots, ou "bots" en abrégé, sont des programmes informatiques qui interagissent avec les gens d'une manière qui ressemble dans une certaine mesure à l'interaction humaine. L'interaction peut varier en complexité, allant de simples requêtes basées sur des mots clés à des systèmes conversationnels élaborés en utilisant le traitement du langage naturel et des techniques d'IA.[2]

Cette forme de conversation d'interaction homme-ordinateur peut servir de cadre pour de nombreux types d'applications utiles. Les robots peuvent simuler une conversation avec un ou plusieurs humains par échange vocal ou textuel.

Les premiers chatbots incluaient des curiosités académiques comme Eliza (1964) et Julia (1994), mais maintenant Internet est inondé de bots qui sont employés dans tout, du service client aux assistants virtuels comme Google Assistant, et dans de nombreux autres rôles.

Un **Chatbot** est donc un logiciel qui interagit avec les utilisateurs en langage naturel via une interface de chat. La conversation peut être textuelle, vocale ou les deux. Les chatbots communiquent avec les utilisateurs pour les aider avec des applications telles que le support client, activités de vente ou de divertissement.

## 2.4 Histoire du chatbot et avènement de l'intelligence artificielle

Historiquement, le premier chatbot appelé Eliza a été créé en 1966 par Joseph Weizenbaum, professeur au MIT (Massachusetts Institute of Technology), aux États-Unis. Le programme, qui simulait un psychothérapeute rogiérien, reformulait la plupart des affirmations de l'orateur en question qu'il lui posait en retour.

A l'origine et pour que le chatbot fonctionne, il s'appuie sur une base de données de questions/réponses. Ces questions réponses sont déclenchées en fonction des mots-clés mentionnés dans la conversation. Cependant, les progrès de l'intelligence artificielle (en particulier l'apprentissage automatique) ont permis d'utiliser de puissants systèmes d'analyse du langage naturel pour créer des agents conversationnels plus avancés, et ils peuvent être améliorés en permanence tels qu'ils sont utilisés.

La figure (2.2) présente l'historique des chatbots en détails :

1. **Google, Facebook et Microsoft ont misé sur les chatbots** : En effet, durant la décennie 2010, les Chatbots ont évolué en valets virtuels capables d'accomplir des tâches et de proposer des solutions en fonction des préférences de la personne qu'ils servent. Apple (avec Siri), Google (avec Google Now), Facebook ainsi que Microsoft (avec Cortana) ont beaucoup investi dans le développement des agents conversationnels pour en faire de véritables assistants. Un autre exemple est l'assistant virtuel Viv. Ils fournissent aux entreprises et aux marques des outils leur permettant de créer des chatbots spécialisés et de les intégrer dans leurs services de messagerie respectifs.
2. **L'agent conversationnel Eugene Goostman** : Un autre chatbot célèbre est Eugene Goostman. Une expérience a eu lieu à Londres en 2014 sous l'égide de la Royal Society of the University of Reading au Royaume-Uni. C'était la première expérience à réussir

le test de Turing de manière convaincante. Cette affirmation semble en fait être exagérée.[3]

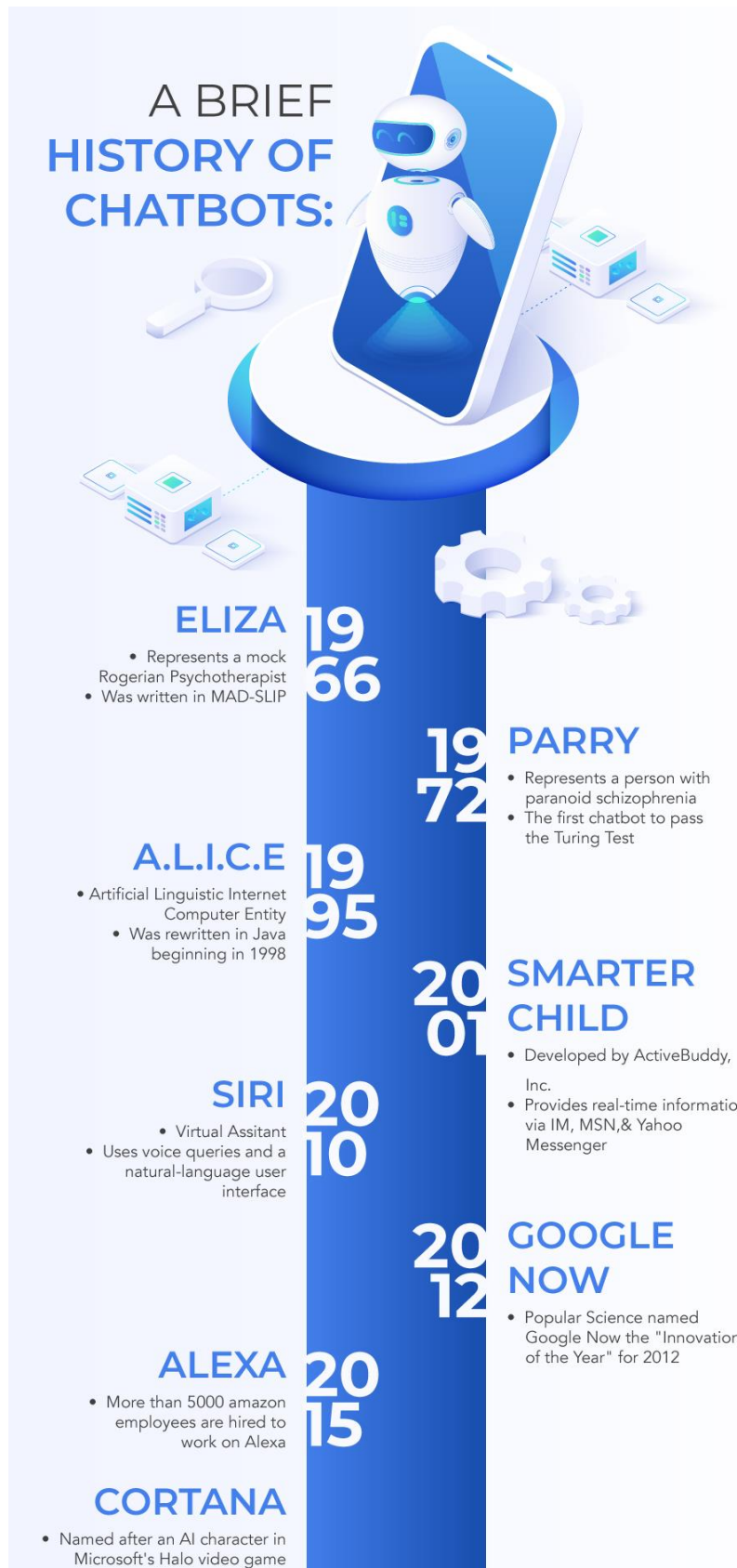


Figure 2.2 : histoire de chatbot

## 2.5 Exemples de chatbots :

Il existe de nombreux exemples de chatbots :

- **Siri** : est l'assistant personnel de l'iPhone qui peut nous aider à trouver le restaurant ou le cinéma le plus proche. Siri peut également suivre notre emploi du temps et vous aider à planifier votre journée.
- **Microsoft Cortana** : Compatible avec diverses plates-formes Microsoft Windows. Il est également compatible avec Android et iOS. Il peut définir des rappels et répondre à des questions en utilisant les informations du moteur de recherche Bing.
- **Assistant Google** : adapté aux appareils domestiques intelligents et aux appareils mobiles. Il fournit les mêmes fonctions que Google : effectue des recherches sur Internet, planifie des événements et des alarmes, ajuste les paramètres matériels sur l'appareil de l'utilisateur et affiche des informations à partir du compte Google de l'utilisateur.
- **Amazon Alexa** : Il permet d'écouter de la musique et des livres audio, conserver des listes de tâches, définissez des alarmes et fournissez des actualités en temps réel et d'autres informations, telles que les prévisions météorologiques, les embouteillages, et l'actualité sportive. Alexa peut également contrôler différents appareils intelligents faisant partie d'un système domotique. Vous pouvez donner des commandes vocales pour contrôler votre système domotique comme "Alexa, éteins la lumière dans le salon".

Le chatbot le plus populaire est le chatbot DoNotPay. Il a été créé par un étudiant de Stanford âgé de 19 ans qui s'appelle Joshua Browder, surnommé "Robin des Bois d'Internet" par la BBC.

## 2.6 Identifier les opportunités pour un chatbot d'Intelligence Artificielle :

La première étape consiste à déterminer la possibilité ou la tâche de résoudre le but et l'utilisation du chatbot. Pour comprendre la meilleure application de Bot au cadre de l'entreprise, vous devez penser aux tâches qui peuvent être automatisées et améliorées avec des solutions d'intelligence artificielle.

Pour chaque type d'activité, les solutions d'intelligence artificielle se répartissent en deux grandes catégories : « Complexité des données » ou « Complexité du travail ». Les deux catégories peuvent être divisées en quatre modèles d'analyse : Rendement, Expertise, Efficacité et Innovation.

## 2.7 Développement de chatbots

Les chatbots deviennent rapidement un incontournable du service client, mais il est important de savoir qu'ils évoluent constamment. L'intégration est l'une des façons la plus intéressante de développer des chatbots. Un chatbot est lui-même une série de réponses programmées, similaire à certains égards à une FAQ interactive. Cependant, l'intégration permet aux chatbots d'accéder aux données historiques pour s'assurer que toutes les questions sont pertinentes pour une requête client spécifique, éliminant ainsi les questions inutiles ou répétitives et créant une expérience plus fluide. L'analyse des sentiments aide également le bot à identifier si un client est déçu et indique que l'agent du centre de contact doit changer. L'entreprise peut également intégrer le rover à des plateformes CRM, des systèmes de communication, des communications unifiées ou des bases de données internes pour optimiser ses avantages et le rendre plus intelligent.[4]

### 2.7.1 Types de chatbots

Il existe de nombreux types de chatbots, dont certains peuvent être grossièrement classés comme suit :

- **Chatbots textuels** : les chatbots textuels utilisent une interface textuelle pour répondre aux questions des utilisateurs.
- **Chatbots vocaux** : avec les chatbots vocaux , le bot répond aux questions des utilisateurs via une interface vocale humaine.

Il existe principalement deux méthodes utilisées pour concevoir les chatbots, décrites comme suit :

- **Approche basée sur des règles** : le robot répond aux questions selon certaines règles apprises. Certaines règles sont très simples ou très complexes. Les robots peuvent gérer des requêtes simples, mais pas des requêtes complexes.
- **Auto-apprentissage** : les bots basés sur l'apprentissage automatique fonctionnent bien mieux que les bots basés sur des règles et sont très efficaces. En plus, ces bots peuvent être classés en deux types : basés sur la demande et génératifs.

En termes de complexité, différents types de chatbots sont disponibles. Certains d'entre eux peuvent être regroupés dans les catégories suivantes.

- **Chatbots traditionnels** : les chatbots traditionnels sont pilotés par les systèmes et l'automatisation. La plupart du temps, cela est fait par des scripts qui ne nécessitent pas beaucoup de travail et des fonctions qui maintiennent simplement le contexte du système.
- **Chatbots en direct** : les chatbots en direct sont alimentés par une communication bidirectionnelle entre les systèmes et les humains.il est capable de gérer les tâches systématiques et le contexte contextuel des tâches.
- **Futurs chatbots** : les futurs chatbots pourront communiquer à plusieurs niveaux grâce à l'automatisation au niveau du système. Sa capacité à prendre en charge les systèmes, les tâches et les environnements des personnes peut déployer un bot principal et éventuellement un système d'exploitation de bot.

### 2.7.2 Principales applications des chatbots :

Il ne fait aucun doute que les chatbots sont largement utilisés dans diverses industries à travers le monde en raison de leurs avantages. Les entreprises existantes et les start-ups espèrent tous utiliser les fonctions des chatbots pour améliorer l'efficacité de tous les aspects de leurs opérations commerciales, telles que l'expérience utilisateur et l'interaction client.

Voici une liste des principales utilisations des chatbots dans diverses entreprises :

- Assistante de réception
- Agent d'assistance
- Tuteur ou enseignant
- Assistant de conduite
- E-mail, réclamations ou distributeur de contenu
- Assistant personnel [exemple : Google Home]

- Assistant aux opérations [exemple : Jarvis du film Iron Maiden]
- Assistant de divertissement [exemple : Amazon Alexa]
- Assistant téléphonique [exemple : Apple Siri]
- Permettre aux personnes malvoyantes de décrire leur environnement
- Peut aider un directeur d'entrepôt à localiser le produit stocké

## 2.8 L'architecture des chatbots :

L'interaction de chat entre le chatbot et l'utilisateur s'effectue par l'échange de messages en langage naturel. L'interaction conversationnelle a un rôle important à jouer car les attentes des clients doivent être gérées tout en leur expliquant comment le chatbot peut les aider.

Une architecture de chatbot nécessite en moyenne les composants suivants

1. Fenêtre de chat/session/ou interface d'application frontale
2. Modèles d'apprentissage en profondeur pour le traitement du langage naturel [NLP]
3. Données pour la formation du modèle PNL
4. Une base de données pour stocker et traiter les actions réalisées par le chatbot.

Voir le (Figure 2.3) pour comprendre l'interface de l'architecture.

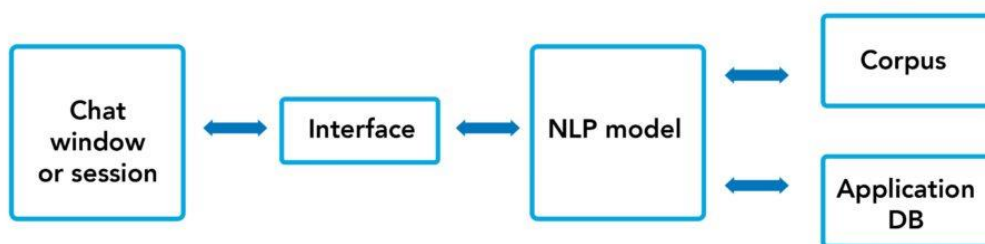


Figure 2.3 : architecture de chatbot

Toutes les solutions ne nécessitent pas la même quantité de conversation, mais sans concevoir consciemment des conversations, le potentiel d'une solution pour répondre aux besoins des clients ne peut pas être réalisé. Si vos clients ne voient pas assez de valeur dans votre solution de chatbot, ils ne s'y tiendront probablement pas. Les objectifs de base de l'interaction conversationnelle sont les suivants :

- Habilitier les utilisateurs et gérer les attentes des utilisateurs.
- Aider à atteindre les objectifs de la solution et ajoute de la valeur.
- Encourager les clients à continuer à utiliser les chatbots.
- Obtenir des informations exploitables sur vos utilisateurs.
- Renforcer la marque du client.
- Augmenter l'empathie et le pardon de l'utilisateur.
- Faire en sorte que votre solution ait l'air intelligente.



Pour atteindre ces objectifs, il faut concevoir soigneusement les interactions afin d'utiliser un langage et des approches appropriés. La conception du chatbot est la partie la plus avancée du développement, et le résultat de cette conception est l'impression générale de l'assistant pour l'utilisateur. Les facteurs clés à prendre en compte lors de la conception d'une conversation sont :

1. Positionnement
2. Ton
3. Personnalité

## 2.9 Chatbots intelligents

### 2.9.1 Chatbots performants

En 2011, le système informatique Watson (Figure 2.4) a concouru sur Jeopardy ! un jeu télévisé diffusé à la télévision américaine contre les anciens vainqueurs Brad Rutter et Ken Jennings. Watson a remporté la première place avec un prix de 1 million de dollars.[6]

IBM Watson a accédé à 200 millions de pages de contenu structuré et non structuré couvrant quatre téraoctets d'espace disque, y compris le texte intégral de Wikipédia.

Le système Watson original qui a remporté Jeopardy ! est basé sur une architecture de système Question-Réponse (QA), il a été démontré en utilisant l'architecture Deep-QA, qui a été développée pour le jeu. Ce système a été spécifiquement développé pour répondre à des questions triviales.



### 2.9.2 Chatbots : à quel point sont-ils intelligents ?

L'âge d'or de l'intelligence artificielle (IA) conversationnelle est arrivé. Les bots conversationnels ou simplement les chatbots ont des applications qui vont des assistants IA en contact avec les clients, des chatbots de support, des chatbots de compétences, des assistants bots et des bots transactionnels. En raison de l'énorme investissement dans cette nouvelle



technologie par le gouvernement, les institutions médicales et les entreprises manufacturières, l'intérêt de l'entreprise dans ce domaine n'est pas limité.

Cependant, les chatbots ne sont pas encore aussi intelligents qu'ils devraient l'être. L'intelligence artificielle continue à s'améliorer, mais les bots semblent manquer d'intelligence émotionnelle. Ils ne peuvent toujours pas fonctionner correctement. De plus, l'expression est complètement différente lorsqu'il s'agit de l'impression d'avoir une conversation avec une personne qui est censée être un bot. C'est pourquoi les bots ne comprennent pas très bien ce qu'on leur demande.

### **2.9.3 Les robots du futur**

Avec de plus en plus d'entreprises utilisant des API pour intégrer des chatbots dans leur écosystème de communication plus large, il existe une demande croissante de bots qui interagissent sur toutes les plateformes. On peut donc attendre à voir une augmentation des SDK et d'autres frameworks qui permettent aux développeurs de créer des robots vocaux pilotés par des API qui basculent de manière transparente entre les canaux, des appels vocaux au chat par message et inversement.

Pendant ces dernières années, la frontière entre ce que les consommateurs attendent de leurs expériences personnelles de communication et ce qu'ils attendent des marques est devenue de plus en plus floue. On attend de plus en plus des entreprises qu'elles s'engagent de la même manière que les clients communiquent avec leurs collègues, amis et même les membres de leur famille. Et, puisque les bots sont principalement alimentés par le traitement du langage naturel et la voix en texte - qui peuvent être appliqués à presque tous les canaux de communication - les marques peuvent désormais offrir aux clients la possibilité de sélectionner leur canal préféré, comme Facebook Messenger, WhatsApp, voix, discuter ou messagerie...

Les chatbots ont révolutionné la façon par laquelle les entreprises interagissent avec leurs clients, et nous n'en sommes qu'au début. Aussi difficile que puisse sembler l'intégration de l'expertise client omnicanale aux dirigeants d'entreprise, elle offre également une excellente opportunité à s'engager avec les clients d'une nouvelle manière, en construisant des relations plus significatives et durables.

## **2.10 Chatbots pour les webinaires**

### **2.10.1 Webinaires**

Un webinaire est un séminaire en ligne dans lequel une conversation peer-to-peer en temps réel est menée d'un expéditeur à plusieurs destinataires.

### **2.10.2 Logiciel/site Web du webinaire**

Cet outil est utile pour tenir des réunions ou des séances d'information. Tant que l'hôte existe, les participants peuvent échanger des applications de bureau et des documents. Le webinaire offre actuellement la possibilité de produire ou de diffuser des enregistrements en direct à l'aide de YouTube et d'autres services vidéo

### **2.10.3 Fonctionnalité de webinaire**

Qu'il s'agisse de séminaires en ligne gratuits, de services financés par la publicité, d'abonnements/payants ou de logiciels, il convient de prendre en compte les fonctionnalités suivantes :

- Plusieurs présentateurs peuvent être pris en charge.
- Les fichiers vidéo peuvent être partagés.

- Le programme peut communiquer avec les participants en temps réel et partager des écrans.
- Options vocales et vidéo préenregistrées. La capacité d'écouter seulement ou de participer à une réunion en direct.
- Filtre pour connecter les participants et les conférenciers sur les médias sociaux.
- Fonctions logicielles interactives de questions et réponses, d'enquêtes et d'outils de rétroaction.
- La capacité de diffusion en continu pour capturer des séquences en direct ou diffuser des vidéos en direct.
- Calendrier des demandes dans le calendrier pour une référence rapide.

#### **2.10.4 Avantages et inconvénients des webinaires :**

Les webinaires dépendent fortement de la technologie, ils ne sont donc pas toujours fiables. Les problèmes techniques tels que la lenteur de la connexion Internet, l'incompatibilité de l'équipement ou des gadgets sont un gros problème. Ce sont les avantages et les inconvénients des webinaires.

##### **1. Avantages :**

- Voyager vers et depuis la destination peut économiser de l'argent et réserver un hôtel la nuit n'est plus un fardeau.
- L'utilisation d'Internet facilite l'inscription.
- Les prospects sont générés à partir de clients potentiels de qualité.
- Les participants sélectionnés peuvent être anonymes.
- Pendant et après la conférence, les informations peuvent être librement échangées.
- Les participants ne sont pas liés par les exigences techniques du téléchargement logiciel de webinaire.
- En tant que matériel numérique, il peut être publié ou téléchargé à tout moment pendant le webinaire.
- Le stockage et l'évaluation du contenu peuvent être faciles.
- L'expertise dans un domaine est augmentée grâce à des webinaires tout en communiquant avec de nouveaux clients et en élargissant vos connaissances en matière d'hébergement Web.

##### **2. Inconvénients :**

- Problèmes techniques pourront survenir pendant la réunion virtuelle, provoquant l'annulation de l'événement ou les participants ne pouvant être présents.
- Il est difficile de déterminer l'humeur et la motivation des participants.
- Distractions et problèmes dans l'environnement, ainsi que le sentiment de ne pas être surveillé, peut être problématique.
- La participation et l'interaction avec le public sont minimales.
- En raison de la période sélectionnée, la connexion est également problématique

#### **2.10.5 Webinaires et chatbots :**

Avant de plonger dans les applications de chatbot, prenons le temps de réfléchir aux facteurs qui permettent aux chatbots d'améliorer le support client. La principale raison du choix des chatbots pour les applications de support client est que les clients préfèrent le texte comme principal moyen de communication avec une marque.

Le service client est une priorité absolue pour toute entreprise - la vérité est que sans avoir des clients satisfaits, vous échouerez bientôt. Mais un engagement client réussi peut s'avérer être un défi à l'ère de la technologie où de nouveaux canaux deviennent constamment disponibles et doivent être intégrés en permanence.

Un assistant pour les webinaires en vaut la peine. Les motifs sont nombreux parmi lesquels, on cite les plus importants :

1. **Clarifiez les préoccupations des participants concernant le logiciel** : sans aucun doute, les nouveaux participants au webinaire ont toujours des questions sur le fonctionnement du logiciel, de la connexion à l'activation du micro en passant par le lancement de l'écran vidéo. Avoir un assistant pour créer un guide ou une vidéo expliquant au public comment faire cela sera inestimable.

Cela peut vraiment aider à la conversion de l'audience, car de nombreuses personnes peuvent éviter d'y aller parce qu'elles sont préoccupées par les plateformes de haute technologie. L'assistant peut gérer toutes les questions que vous pourriez avoir sur ce processus. Cela peut être très utile car cela évite autant de problèmes que possible lors de l'événement réel pour une présentation fluide.

2. **Combattre facilement les problèmes techniques** : les problèmes techniques semblent survenir aux pires moments possibles et sont généralement un peu plus complexes que prévu. Avoir un assistant de garde est indispensable pour résoudre les problèmes techniques - et d'autres problèmes en arrière-plan. Trop de webinaires ont échoué parce que l'hôte n'est pas en mesure de résoudre un problème. Les gens attendront un certain temps, mais si cela prend trop de temps pour résoudre le problème, ils se déconnecteront du webinaire et reprendront d'autres responsabilités. De plus, il n'y a rien de pire que le présentateur ait à faire face à de tels problèmes car cela peut facilement lui faire perdre le focus de la présentation elle-même.
3. **Séance modérée de questions et réponses** : les séances de questions et réponses constituent une partie importante d'un webinaire, car elles offrent aux participants la possibilité de poser des questions ou d'exprimer leurs préoccupations concernant les informations présentées.

Étant donné que les webinaires sont limités dans le temps, répondre aux questions peut facilement devenir incontrôlable et interférer avec la gestion du temps, et c'est là qu'un assistant peut être un atout inestimable. Il y a plusieurs avantages à utiliser un assistant pour modérer les questions du public. Il ou elle peut sélectionner les questions auxquelles le présentateur répondra pendant la diffusion en direct, filtrer les doublons et enregistrer les questions sans réponse pour une utilisation future, par exemple pour que le présentateur réponde dans un e-mail de suivi. L'assistant peut traiter les questions selon un temps et un format prédéfinis (tels que déterminés par le présentateur), puis les trier et les organiser en conséquence. Les participants sont souvent frustrés lorsque le modérateur ne garde pas la conversation ciblée et ne respecte pas l'horaire. Trouver un assistant qui est un modérateur qualifié ajoutera vraiment de la valeur à votre présentation de webinaire.

4. **Professionalisme du projet** : L'utilisation d'un assistant pour modérer les questions, répondre aux besoins de votre public et résoudre les problèmes techniques ajoutera du professionnalisme à votre présentation. Vous aurez l'air très organisé et préparé. De plus, cela peut faire paraître votre entreprise plus grande, avec du personnel supplémentaire pour gérer les présentations et résoudre les problèmes informatiques.
5. **Permet au présentateur de se concentrer sur le contenu** : L'un des avantages les plus importants d'un assistant est qu'il permet au présentateur de se concentrer sur l'aspect le plus important du webinaire, à savoir le contenu et la présentation elle-même. Tous les problèmes techniques et de modération étant gérés par un assistant compétent, le présentateur peut consacrer son temps et son énergie à la création et à la présentation effective du contenu, qui, après tout, est l'aspect le plus important de tout webinaire.

## 2.11 Cadre Rasa

### 2.11.1 Définition

**Rasa** est un framework d'apprentissage automatique open source pour la communication textuelle et vocale automatisée. Comprenez les messages, tenez des conversations et connectez-vous aux canaux de messagerie et aux API.[6]

### 2.11.2 Explorer Rasa

En utilisant Rasa, vous pouvez créer des chatbots avec trois composants principaux :



Figure 2.4 : logo de rasa

### Rasa NLU ((Natural Language Understanding))

Rasa NLU est comme "l'oreille" d'un chatbot, il aide l'assistant à comprendre ce qui se dit. Rasa NLU accepte les entrées de l'utilisateur sous la forme d'un langage humain non structuré et récupère les données structurées sous la forme d'intentions et d'entités.

- **Intentions** : en général, il représente le but ou l'objectif de l'entrée d'un client, ce sont des étiquettes qui représentent le but ou la signification de l'entrée particulière d'un utilisateur. Par exemple, le message 'hello' peut avoir l'étiquette 'greet' car il implique une salutation.
- **Entités** : fait référence globalement aux termes ou objets liés à l'intention de l'utilisateur et fournit un contexte spécifique pour l'intention. Ce sont des mots-clés importants auxquels l'assistant doit prêter attention.

Par exemple, le message "Je m'appelle Dhekra" contient le nom Dhekra. Un assistant doit ici extraire le nom et s'en souvenir tout au long de la conversation pour que l'interaction reste naturelle.

## Rasa core

La gestion du dialogue est le composant central de Rasa. Il décide comment le chatbot doit répondre en fonction de l'état de la conversation et du contexte. Rasa Core apprend en observant des modèles dans les données de conversation entre les utilisateurs et l'assistant.

## Rasa X

Rasa X est un ensemble d'outils permettant aux programmeurs de créer, d'améliorer et de déployer des assistants à l'aide du framework Rasa. Rasa X peut être utilisé pour :

- Visualisez et annotez facilement les conversations
- Demandez aux testeurs leur avis.
- Gérer les modèles et les versions"

Avec Rasa X, nous pouvons partager notre chatbot avec de vrais testeurs et collecter les conversations qu'ils ont avec l'assistant, ce qui nous permet d'améliorer notre bot sans interrompre l'exécution de l'assistant en production.[7]

### 2.11.3 Structure du projet Rasa

La commande "rasa init" crée un nouveau projet Rasa dans un répertoire local que nous fournissons en spécifiant le nom du répertoire. Dès que le répertoire est initialisé, Rasa le remplit automatiquement avec les fichiers de projet et les exemples de données de formation et forme les modèles NLU et de dialogue.

Par défaut, "rasa init" forme un simple chatbot appelé Moodbot qui vous demande comment vous vous sentez. Lorsque nous sommes contrariés, il vous remontera le moral en vous envoyant des photos de mignons petits tigres.

### *Données d'entraînement NLU*

Un répertoire de données est inclus dans le projet de démarrage Moodbot où nous pouvons trouver les fichiers de données de formation pour les modèles de gestion NLU et Dialog. Le répertoire Data contient deux fichiers :

- **stories.yml** : Ce fichier contenant les données de story. Un exemple de conversation de bout en bout est une story.
- **nlu.yml** : le fichier contient des exemples de formation de modèle NLU. Cela contient des intentions, qui sont des objectifs de l'utilisateur, et des énoncés qui illustrent ces intentions. Les données de formation NLU identifient également les entités ou les mots-clés clés que le chatbot doit extraire de l'exemple d'énoncé.

### *Gestion du dialogue avec Rasa*

La gestion du dialogue est la fonction qui contrôle la prochaine action de l'assistant lors d'une connexion. Sur la base des intentions et des entités extraites par Rasa NLU, ainsi que d'autres contextes, comme l'historique des discussions, Rasa core décide quelle réponse textuelle doit être renvoyée à l'utilisateur ou s'il faut exécuter un script personnalisé tel qu'une recherche dans la base de données.

- **domaine.yml** : Le domaine est une partie essentielle d'un modèle de gestion de dialogue Rasa. Il représente l'environnement global dans lequel le bot s'exécute, notamment :
  - Ce que l'utilisateur veut dire : plus précisément, quelles intentions et quelles entités peuvent être comprises par le modèle.
  - Quelles réponses le modèle peut fournir : comme des énoncés ou des actions personnalisées.
  - Que dire ensuite : à quoi le modèle doit-il être prêt à répondre ?
  - À quelles informations faut-il prêter attention ? Quelles informations l'assistant doit-il enregistrer et utiliser pendant la conversation ?

Dans ce fichier de domaine, nous pouvons voir cinq parties qui sont nécessaires pour créer un assistant avec ce framework : les intentions, les entités, les slots, les actions et les modèles/réponses.

- **Intents** : La section nommée intents définit une liste d'intents que le chatbot est capable de comprendre. Ces détails proviennent du modèle NLU (nlu.yml). La section des intentions est l'endroit où nous devons fournir les étiquettes de toutes les intentions que nous avons entraîné notre modèle NLU à comprendre.
- **Actions** : la section intitulée actions doit contenir la liste de tous les énoncés et actions personnalisées qu'un chatbot doit utiliser pour répondre aux entrées de l'utilisateur. Celles-ci doivent provenir des données des histoires dans le fichier stories.yml.
- **Entités** : une autre section importante du fichier de domaine concerne les entités. Le domaine de l'assistant Moodbot manque cette partie. Étant donné que les entités influencent la façon dont un assistant répond à l'entrée d'un utilisateur, les entités doivent être présentées dans le fichier domain.
- **Slots** : les fichiers de domaine contiennent également des slots, qui sont très importants pour la gestion des dialogues dans Rasa. Les emplacements fonctionnent comme la mémoire de l'assistant et sont utilisés par l'assistant pour enregistrer et mémoriser des détails importants pendant la conversation et mettre ces détails en pratique dans le contexte pour conduire la conversation. Les emplacements sont utilisés comme paires clé-valeur pour stocker des informations importantes pour les conversations avec les utilisateurs. Ces informations peuvent être fournies par l'utilisateur (par exemple, la valeur d'entité obtenue à partir du modèle NLU) ou collectées en dehors de la conversation (par exemple, le résultat obtenu à partir d'une base de données externe).
- **Templates Responses** : La dernière section du domaine s'appelle les réponses. La partie des réponses est l'endroit où nous pouvons définir les réponses textuelles spécifiques que le chatbot renverra à l'utilisateur, en fonction de l'énoncé prédit par le modèle de gestion du dialogue. Un seul énoncé peut avoir plusieurs réponses, c'est-à-dire plusieurs réponses textuelles qui capturent avec précision le sens de l'énoncé. Les énoncés peuvent contenir non seulement des messages simples, mais également des images, des boutons, des charges utiles personnalisées, etc.

- **Actions.py** : ajouter des modèles de réponse directement au fichier de domaine est le moyen le plus simple de définir le message qu'un assistant envoie à l'utilisateur lorsqu'une expression spécifique est attendue. Mais il est également possible d'obtenir le même résultat en créant des actions personnalisées.
- **config.yml** : ce document contient le chatbot du pipeline de formation. Le pipeline de formation utilise une série d'étapes de traitement pour former un nouveau modèle NLU qui permet au modèle d'apprendre des modèles de données de formation de base.

Comme nous pouvons le constater, il existe un lien évident entre le domaine, les données d'entraînement NLU et les fichiers de données stories. Il n'y a pas de règle spécifique pour laquelle on devrait venir en premier, mais nous remarquerons que les changements dans un fichier entraîneront des changements dans d'autres fichiers.

## 2.12 Chatbots à commande vocale

Ajouter une voix à un chatbot le rend plus interactif et valorise sa personnalité.

Cela ne nécessite presque aucun effort car la plupart des services de synthèse vocale peuvent être ajoutés au texte de réponse de sortie existant pour le transformer en audio.

### 2.12.1 Parole en texte

STT est un outil de technologie d'assistance qui permet aux enfants qui ont du mal à écrire, aux personnes handicapées ou même aux personnes normales d'interagir plus facilement avec les machines. La dictée est également appelée technologie parole-texte, voix-texte, reconnaissance vocale et reconnaissance vocale. Avec cette technologie, les mots parlés (avec le langage humain) deviennent du texte numérique sur un écran.

#### Outils STT

Python prend en charge la reconnaissance vocale et est compatible avec de nombreux packages de reconnaissance vocale open source. Il existe de nombreux packages de reconnaissance vocale Python disponibles à ce jour. Voici quelques-uns des plus populaires :

- Apiai
- Google-cloud-discours
- Parole IBM au texte
- Reconnaissance de la parole

### 2.12.2 Synthèse vocale

Text-to-speech (TTS) est un type de technologie qui aide cette technologie à lire le texte numérique à haute voix. On parle parfois de technologie de « lecture à haute voix ».

#### Bases de la synthèse vocale

Un large éventail de fichiers texte, y compris des documents Word et Pages, ainsi que des pages de sites Web peuvent être lus à haute voix à l'aide de TTS. Il fonctionne avec presque tous les appareils numériques personnels, y compris les smartphones, les tablettes et les ordinateurs.

Dans TTS, les voix sont générées par ordinateur et la vitesse de lecture peut varier. La qualité des voix peut aller de la voix machine à la voix humaine. Il y a même des voix qui ressemblent à des enfants qui parlent.

## Outils TTS

Pytttsx3 est une bibliothèque de conversion texte-parole en Python. Contrairement aux bibliothèques alternatives, elle fonctionne hors ligne et est compatible avec Python 2 et 3.

Le module pytttsx3 prend en charge deux voix, la première est féminine et la seconde est masculine, fournie par « sapi5 » pour Windows. Et il prend en charge trois moteurs TTS :

- SAPI5 sous Windows
- NSSpeechSynthesizer sur Mac OS X
- eSpeak sur toutes les autres plates-formes

### 2.12.3 Structure d'une conversation vocale

Voici l'architecture globale (Figure 2.6) d'une interaction vocale avec un chatbot et comment se déroule la compréhension du message et comment obtenir une réponse claire du bot.

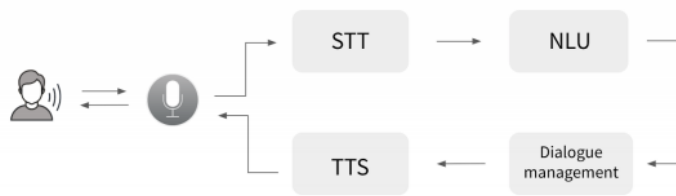


Figure 2.5 : les étapes d'une conversation vocale

## 2.13 Conclusion

Auparavant, les applications de messagerie n'étaient utilisées que pour discuter avec des amis. Mais avec l'essor de l'intelligence artificielle, les entreprises peuvent désormais converser automatiquement avec leurs clients grâce aux bots. Les chatbots ont le vent en poupe, et c'est le nouveau mot à la mode.

La fonction "rasa init" est un excellent moyen de voir comment fonctionne un chatbot alimenté par Rasa. Nous pouvons également utiliser Moodbot comme projet passe-partout pour créer notre propre assistant personnalisé.

Tout au long de ce chapitre, nous regardons ce qu'est un chatbot, nous décrivons les applications courantes des chatbots. Nous énumérons de nombreux avantages de l'utilisation des chatbots et pourquoi nous utiliserons des chatbots et des assistants personnels.



## Chapitre 3

### 3 Réalisation Partie 1 : Créer un Chatbot avec Rasa

#### 3.1 Introduction

Ce chapitre aborde les aspects techniques liés à l'intégration et à la réalisation du projet. Nous présentons à la fois les environnements matériels et logiciels destinés à la mise en œuvre de notre application. Nous exposons les différentes phases de création de l'assistant ainsi qu'un aperçu de l'intégration de la base de données.

#### 3.2 Environnement de travail

##### *Environnement physique*

L'ordinateur portable utilisé lors de la réalisation du projet présente les caractéristiques suivantes :

##### Device specifications

|               |   |
|---------------|---|
| Device name   | DESKTOP-QPL0KG8                                     |
| Processor     | Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz   |
| Installed RAM | 8.00 GB (7.89 GB usable)                            |
| Device ID     | 84F142AD-15B9-44D6-A049-10C67FAAE54E                |
| Product ID    | 00331-10000-00001-AA894                             |
| System type   | 64-bit operating system, x64-based processor        |
| Pen and touch | No pen or touch input is available for this display |

Figure 3.1 : les caractéristiques de pc

##### 3.2.1 Environnement de développement

Cette section contient une liste d'outils logiciels (Figure 3.2) que nous avons utilisés lors de la réalisation de l'application.



Figure 3.2 : environnement logiciel

## **Docker**

Il s'agit d'un outil permettant de créer, de déployer et d'exécuter des applications à l'aide de conteneurs. À l'aide de conteneurs, les développeurs peuvent emballer des applications avec tous les composants nécessaires, tels que des bibliothèques et d'autres dépendances, puis les déployer en tant que package unique. Ce faisant, le développeur peut s'assurer que l'application peut s'exécuter sur n'importe quel autre ordinateur Linux, quels que soient les paramètres utilisateur sur cet ordinateur, qui peuvent être différents de l'ordinateur sur lequel le code est écrit et testé. [8]

## **Rasa**

Un cadre d'apprentissage automatique open source, Rasa a automatisé les conversations vocales et textuelles grâce à l'apprentissage automatique. Tenir des conversations, comprendre les messages et se connecter aux canaux de messagerie et aux APIs

## **MongoDB**

MongoDB est une base de données de documents avec l'évolutivité et la flexibilité que vous souhaitez avec l'interrogation et l'indexation dont vous avez besoin. Il s'agit d'une base de données distribuée en son cœur, de sorte que la haute disponibilité et la distribution géographique sont intégrées et faciles à utiliser. [9]

## **Code Visual Studio**

Est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, mac OS et Linux. Il est livré avec un support intégré pour TypeScript, JavaScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages (tels que C++, C#, Java, Python, PHP, Go) et des runtimes (tels que .NET et Unity).[10]

## **Python**

Python est un langage de programmation de haut niveau, interprété, orienté objet et doté d'une sémantique dynamique. Sa construction de haut niveau dans les structures de données, combinée au typage dynamique et à la liaison dynamique, le rend très attrayant pour le développement rapide d'applications, ainsi que pour une utilisation en tant que langage de script pour connecter des composants existants entre eux. [11]

## **Angular**

Angular est un cadre de conception d'applications et une plate-forme de développement permettant de créer des applications d'une seule page efficaces et sophistiquées. [12]

### **3.2.2 Environnement en ligne**

Ce projet a également besoin de certaines plates-formes pour la gestion des versions, les tests et le déploiement (Figure 3.3).

## **Microsoft Azure**

Il s'agit d'un service de cloud computing créé par Microsoft pour développer, tester, déployer et gérer des applications et des services via des centres de données gérés par Microsoft. Il fournit IaaS (Infrastructure as a Service), PaaS (Platform as a Service) et SaaS (Software as a Service),

et prend en charge une grande variété de langages de programmation, d'outils et de cadres, y compris Logiciels et systèmes spécifiques à Microsoft et tiers. [13]  
Azure for Students vous accorde 100 \$ de crédit et peut utiliser plus de 25 produits gratuits dans les 12 mois, y compris le calcul, les réseaux, le stockage et les bases de données.



*Figure 3.3: environnements virtuels*

### **DockerHub**

Docker Hub est la plus grande bibliothèque et communauté d'images de conteneurs au monde. Parcourir plus de 100 000 images de conteneurs provenant de fournisseurs de logiciels, de projets open source et de la communauté. [14]

### **GitHub GenericName**

Fournit des services d'hébergement pour le contrôle de version et le développement de logiciels à l'aide de Git. Il offre la fonctionnalité de gestion du code source (SCM) et le contrôle de version distribué de Git, ainsi que ses propres fonctionnalités. [15]

### **Héroku**

Est une plate-forme basée sur le cloud qui permet aux entreprises de créer, de fournir, de surveiller et de mettre à l'échelle des applications. Héroku est une plate-forme cloud en tant que service basée sur des conteneurs. Des plates-formes comme Héroku aident les développeurs à implémenter, gérer et faire évoluer des applications modernes. [16]

## **3.3 Modèle NLU**

Dans un projet d'apprentissage automatique, nous avons besoin d'un ensemble de données d'entraînement, qui est un ensemble de données réel utilisé pour entraîner le modèle à effectuer diverses actions.

### 3.3.1 Compréhension du langage naturel

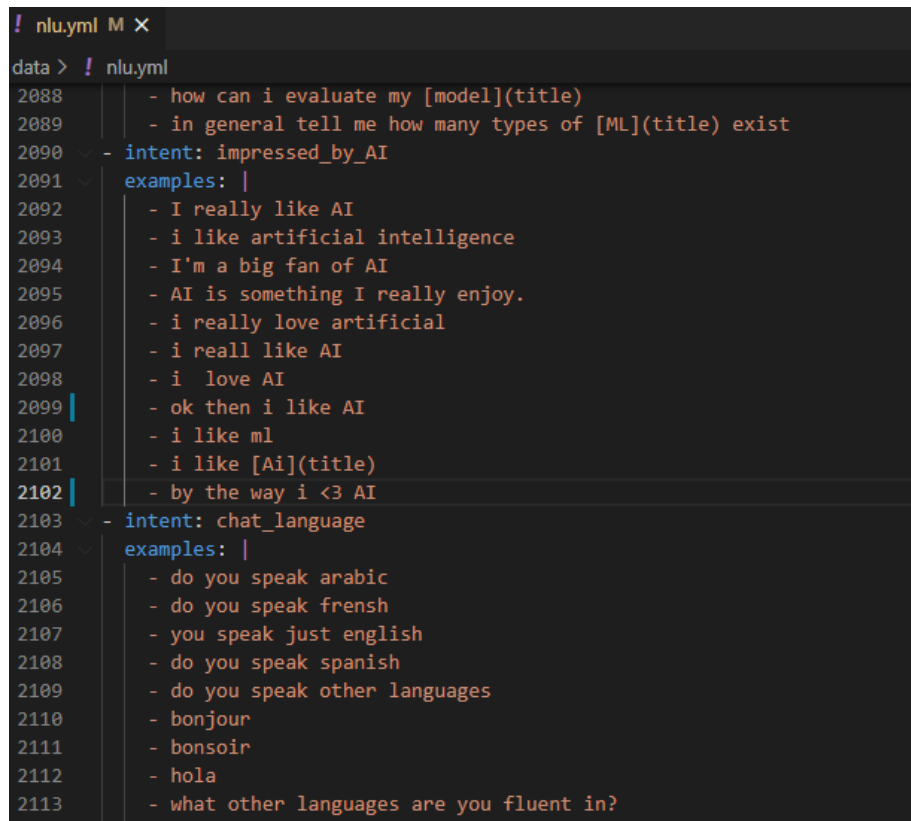
Natural Language Understanding NLU est une catégorie de traitement du langage naturel qui modélise la compréhension de la lecture humaine, ou en d'autres termes, analyse et interprète les entrées en utilisant les principes du langage naturel. La NLU est considérée comme un problème difficile en intelligence artificielle.[17]

### 3.3.2 Génération des données de formation NLU pour notre chatbot

Chaque intention que l'assistant soit capable de comprendre devra être définie dans le fichier nlu. Il y a quelques bonnes pratiques à garder à l'esprit :

- Nous n'avons pas besoin d'écrire tous les énoncés possibles pour former une intention, mais nous devrions fournir 10 à 15 exemples.
- Pour former le modèle, nous avons besoin de données de haute qualité. Les exemples utilisés doivent correspondre aux intentions.

Dans la figure (Figure 3.4), l'intention imprimée par l'IA est suivie de plusieurs exemples de la manière dont un utilisateur pourrait exprimer son attirance pour le domaine de l'IA.



```
! nlu.yml M X
data > ! nlu.yml
2088 - how can i evaluate my [model](title)
2089 - in general tell me how many types of [ML](title) exist
2090 - intent: impressed_by_AI
2091   examples: |
2092     - I really like AI
2093     - i like artificial intelligence
2094     - I'm a big fan of AI
2095     - AI is something I really enjoy.
2096     - i really love artificial
2097     - i reall like AI
2098     - i love AI
2099     - ok then i like AI
2100     - i like ml
2101     - i like [Ai](title)
2102     - by the way i <3 AI
2103 - intent: chat_language
2104   examples: |
2105     - do you speak arabic
2106     - do you speak frensh
2107     - you speak just english
2108     - do you speak spanish
2109     - do you speak other languages
2110     - bonjour
2111     - bonsoir
2112     - hola
2113     - what other languages are you fluent in?
```

Figure 3.3 : génération de données de formation NLU

Afin d'obtenir un modèle NLU performant, nous devons introduire de grandes quantités de données (intentions et entités) pour former le modèle. Ce n'est pas pratique à faire manuellement, nous allons donc utiliser des bots (scripts python) qui peuvent nous aider à créer nos données nlu rapidement et sans fautes de frappe.

La figure (figure 3.5) présente un bot qui va générer les exemples d'intent "getting-ai-concepts-definition ".

### 3.3.3 Pipeline de formation NLU

Une fois que nous avons créé nos données d'entraînement, nous sommes prêts à configurer le pipeline, qui formera un modèle sur ces données. Le pipeline de traitement de l'assistant est défini dans le fichier `config.yml`, qui est généré automatiquement lorsque nous créons un projet de démarrage à l'aide de la commande `rasa init`. Dans cette partie, nous nous concentrerons sur le choix de la bonne configuration de pipeline de formation.

#### Canal de formation (training pipeline) :

Les modèles NLU sont créés par un pipeline de formation, également appelé pipeline de traitement.

Un pipeline de formation est une séquence d'étapes de traitement qui permet au modèle d'apprendre les modèles sous-jacents des données de formation. Chaque étape du pipeline est exécutée en séquence et l'ordre des étapes est très important.

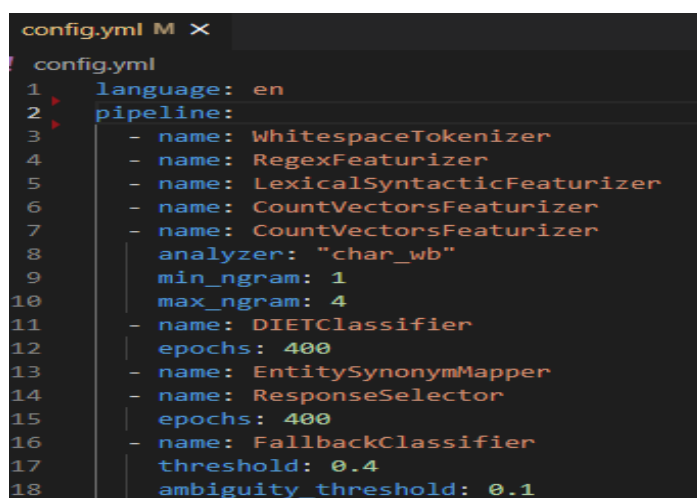
### 3.3.4 Choix d'une configuration de pipeline :

Rasa est livré avec deux pipelines préconfigurés par défaut :

- **Pretrained embeddings spacy** : il utilise le modèle de langage pré-entraîné spacy.
- **Supervised embeddings** : il entraîne le modèle à partir de zéro en utilisant les données fournies dans le fichier de données d'entraînement NLU.

Il est possible d'effectuer une classification d'intention et une extraction d'entité avec les deux pipelines. Dans notre cas, nous choisirons les intégrations supervisées car nous avons des termes spécifiques (termes arabes tels que les noms d'individus, certains termes scientifiques, etc.). Mais nous personnalisons notre modèle car les pipelines préconfigurés sont un excellent moyen pour commencer rapidement, mais à mesure que le projet devient plus complexe, nous voudrions probablement personnaliser le modèle.

[Supervised embeddings](#) contiennent de nombreux composants et ces composants constituent notre pipeline NLU et fonctionnent de manière séquentielle pour traiter les entrées de l'utilisateur en une sortie structurée. Il existe des composants pour l'extraction d'entités, la classification des intentions, la sélection des réponses, le prétraitement, etc.



```
config.yml M X
? config.yml
1 language: en
2 pipeline:
3   - name: WhitespaceTokenizer
4     - name: RegexFeaturizer
5     - name: LexicalSyntacticFeaturizer
6     - name: CountVectorsFeaturizer
7     - name: CountVectorsFeaturizer
8       analyzer: "char_wb"
9       min_ngram: 1
10      max_ngram: 4
11     - name: DIETClassifier
12       epochs: 400
13     - name: EntitySynonymMapper
14     - name: ResponseSelector
15       epochs: 400
16     - name: FallbackClassifier
17       threshold: 0.4
18     ambiguity_threshold: 0.1
```

Figure 3.4 : configuration de pipeline

### 3.3.5 Composants du pipeline de formation

La figure (3.6) présente notre pipeline choisi, nous devons donc examiner chaque composant de ce pipeline plus en détail.

1. **White Space Tokenizer** : Il recherche les espaces blancs dans un flux de texte et les utilise comme délimiteur pour séparer chaque jeton.
2. **Regex Featurizer** : le composant peut être ajouté avant le composant extracteur d'entité pour faciliter l'extraction d'entité lors de l'utilisation d'expressions régulières et de tables de recherche.
3. **Caractéristique syntaxique lexicale** : crée des fonctionnalités lexicales et syntaxiques pour un message utilisateur afin de prendre en charge l'extraction d'entités.
4. **Count Vectors Featurizer** : Cette fonctionnalité crée une représentation de sac de mots du message d'un utilisateur à l'aide de CountVectorizer de sklearn. Le modèle de sac de mots ignore l'ordre des mots dans un corps de texte et se concentre plutôt sur le nombre de fois que les mots apparaissent dans le texte. Ainsi, ce composant compte la fréquence à laquelle certains mots de vos données d'entraînement apparaissent dans un message et les fournit comme entrée pour le classificateur d'intention.
5. **DIET (Dual Intent and Entity Transformer)** : est une architecture multitâche pour la reconnaissance d'entités et la classification d'intentions. Il utilise un transformateur qui est partagé entre les deux tâches. La séquence d'étiquettes d'entité est prédite en incorporant une couche de marquage de champ aléatoire conditionnel (CRF) sur la séquence de sortie du transformateur qui correspond à la séquence d'entrée de jetons. Dans le cas des étiquettes d'intention, la sortie du transformateur pour l'énoncé complet et les étiquettes d'intention sont combinées en un seul espace vectoriel sémantique. De plus, pour ajuster le modèle, nous pouvons définir des hyperparamètres. [18]  
Si on veut adapter le modèle, il faut commencer par modifier le paramètre suivant :
  - Epochs : ce paramètre définit le nombre de fois où l'algorithme verra les données d'apprentissage.
6. **Entity Synonym Mapper** : si les données d'apprentissage contiennent des synonymes définis, ce composant s'assurera que les valeurs d'entité détectées seront mappées à la même valeur.
7. **Sélecteur de réponse** : il peut être utilisé pour créer un modèle de récupération de réponse afin de prédire directement une réponse de bot à partir d'un ensemble de réponses candidates.

8. **Classificateur de secours** : classe un message utilisateur avec l'intention nlu-fallback au cas où le classificateur d'intention précédent n'a pas été en mesure de classer une intention avec une confiance supérieure ou égale au seuil du classificateur de secours. Il peut également prédire l'intention de repli lorsque les valeurs de confiance des deux intentions les mieux notées sont plus proches que le seuil d'ambiguïté.

Pour finir avec cette partie, la question est : L'ordre des composants dans le pipeline est-il important ?

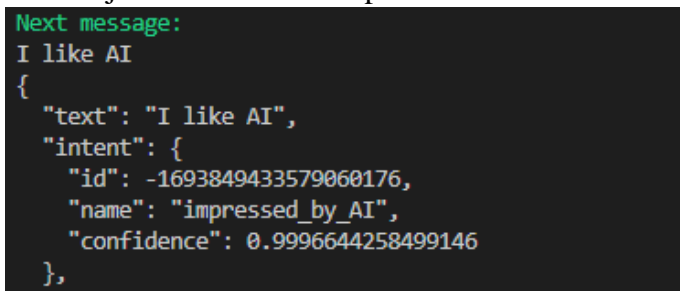
La réponse courte est oui ! Certains composants ont besoin de la sortie d'un composant précédent pour faire leur travail. En règle générale, le Tokenizer doit être au début du pipeline et le featurer doit venir avant le classificateur d'intention.

### 3.3.6 Test du modèle

Essayer le modèle nouvellement formé en utilisant "rasa shell nlu", une commande fournie par Rasa CLI. Il charge le dernier modèle NLU et vous permet de tester ses performances en conversant avec l'assistant sur la ligne de commande.

En mode test, tapez un message dans votre terminal par exemple, "I like IA" (Figure 3.7)

Rasa CLI génère un objet JSON contenant plusieurs éléments de données utiles :



```
Next message:
I like AI
{
  "text": "I like AI",
  "intent": {
    "id": -1693849433579060176,
    "name": "impressed_by_AI",
    "confidence": 0.9996644258499146
  },
  "fallback_intent": "nlu_fallback"
```

Figure 3.5 : tester le modèle

L'intention que le modèle devine est la plus susceptible d'être représentée par le message. Cela signifie que le modèle est certain à 99 % que "I like AI" est un proud\_by\_AI. À l'aide de cette sortie, nous pouvons comparer les performances des modèles générés par différentes configurations de pipeline.

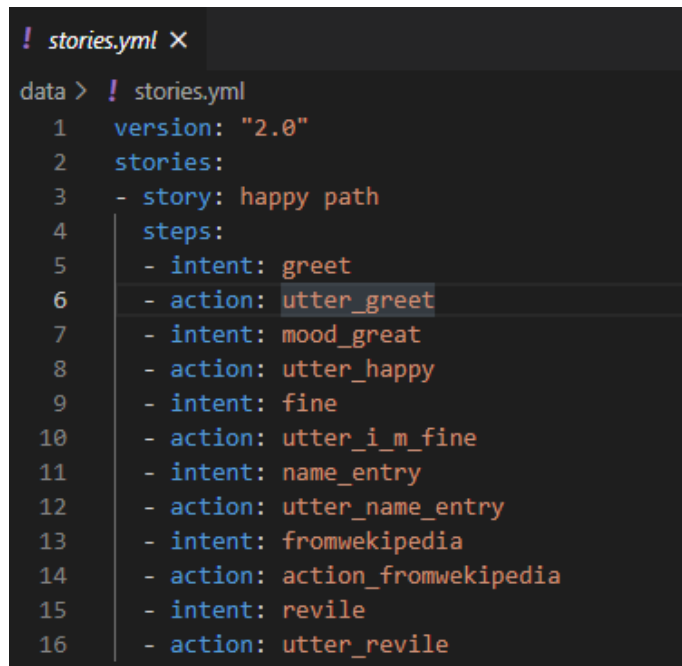
## 3.4 Gestion des dialogues

Le noyau Rasa utilise l'apprentissage automatique pour capturer des modèles conversationnels à partir d'exemples conversationnels. Ces exemples de dialogue sont fournis en tant que données de formation, tout comme la méthode que nous utilisons pour former le modèle NLU. Par conséquent, l'assistant peut généraliser, permettant au modèle d'identifier la meilleure action suivante à entreprendre au cours d'une conversation. De cette façon, même si le dialogue ne correspond à aucun des exemples d'entraînement évoqués ci-dessus, le modèle peut donner des réponses appropriées.

### 3.4.1 Stories :

L'unité de base de la formation au dialogue dans Rasa core est une story. Nous pouvons considérer une story comme un script détaillant la conversation entre l'utilisateur et l'assistant, du début à la fin. Les stories sont écrites dans un format spécifique et stockées dans le fichier

des stories. Une fois la commande `rasa init` exécutée pour créer un nouveau projet de démarrage, le fichier `stories.yml` est automatiquement placé dans le répertoire de données, et plusieurs stories de formation simple seront incluses. Nous allons commencer par une story(histoire) dans laquelle les utilisateurs fournissent toutes les informations dont ils ont besoin, et la conversation se déroule comme prévu. (Ce que nous appellerions le "chemin heureux" ("happy path")). Examinons (Figure 3.8) les fichiers de récits et le format de récit de formation.



```
! stories.yml X
data > ! stories.yml
1  version: "2.0"
2  stories:
3  - story: happy path
4    steps:
5    - intent: greet
6      - action: utter_greet
7    - intent: mood_great
8      - action: utter_happy
9    - intent: fine
10     - action: utter_i_m_fine
11     - intent: name_entry
12     - action: utter_name_entry
13     - intent: fromwikipedia
14     - action: action_fromwikipedia
15     - intent: revile
16     - action: utter_revile
```

Figure 3.6 : Rasa core : Stories

Tout d'abord, le nom de l'histoire ou de story est un chemin heureux (« happy path »). Ensuite, et parce que l'utilisateur commencera probablement par une salutation comme "hello" ou "hi", nous commençons l'histoire avec l'intention de salutation de l'utilisateur, qui est une intention prédéfinie lorsque nous créons les données d'entraînement NLU de l'assistant.

En réponse, l'assistant répond par un énoncé : un message codé en dur qui imprime "Hi. I am a Sara. I can assist you in AI domain, webinars details and acarobotics information." et ainsi de suite jusqu'à ce que l'utilisateur remercie l'assistant et que l'assistant prononce au revoir.

### 3.4.2 Domaine, actions personnalisées et emplacements(slots)

#### Fichier de domaine dans Rasa

Nous avons défini et expliqué les fichiers de domaine au chapitre 3, ces fichiers contiennent cinq sections qui sont nécessaires pour construire un assistant avec Rasa. Voici donc l'endroit où nous définissons nos : intentions, actions et réponses, entités et créneaux. (Intents, actions and responses, entities and slots)



Examinons de plus près notre fichier de domaine de chatbot (Figure 3.9), précisément la section des réponses.

```
! domain.yml M X
! domain.yml
96 responses:
97   utter_greet:
98     - text: Hey ! How are you?
99     - text: Hey friend! How are you?
100    - text: Hi ! How are you?
101    - text: Hi there! how are you ?
102   utter_name_entry:
103     - text: nice to meet you . How can I help you {name} ?
104     - text: ' It is a pleaser to speak with you.Can I help you {name} !!! '
```

Figure 3.7 : Rasa core : Domain File

La section de réponse ou de modèle est l'endroit où nous pouvons définir les messages spécifiques que le chatbot fournira à l'utilisateur, en fonction de la prononciation prédite par le modèle de gestion du dialogue.

Ici, par exemple, lorsque l'utilisateur saisit un exemple d'intention de salutation(greeting), le chatbot prendra un texte de utter\_greet renvoyez-le à l'utilisateur. Nous mettons beaucoup de texte pour permettre à notre chatbot de répondre avec des réponses diversifiées (si l'utilisateur dit bonjour deux fois, la deuxième réponse doit être différente de la première).

Un autre élément important que nous pouvons voir ici est le slot (En fait, dans le texte "nice to meet you. How can I help you name", the name is a slot). Au début de la conversation, le bot peut demander à l'utilisateur nom, puis mémorisez ce nom pour rendre la conversation plus conviviale et plus humaine.

### Actions personnalisées dans Rasa

Les actions personnalisées sont des actions de réponse qui incluent du code personnalisé. Ce code personnalisé peut définir n'importe quoi, d'une simple réponse textuelle à une intégration back-end - un appel API, une connexion à la base de données (figure 3.10) ou toute autre chose que le chatbot doit faire.

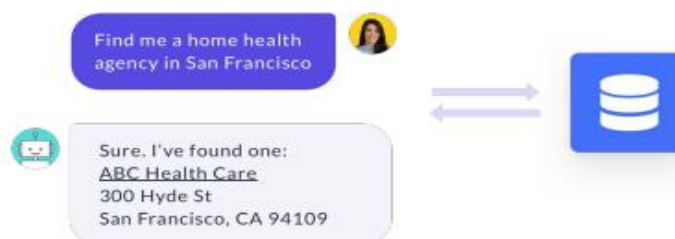
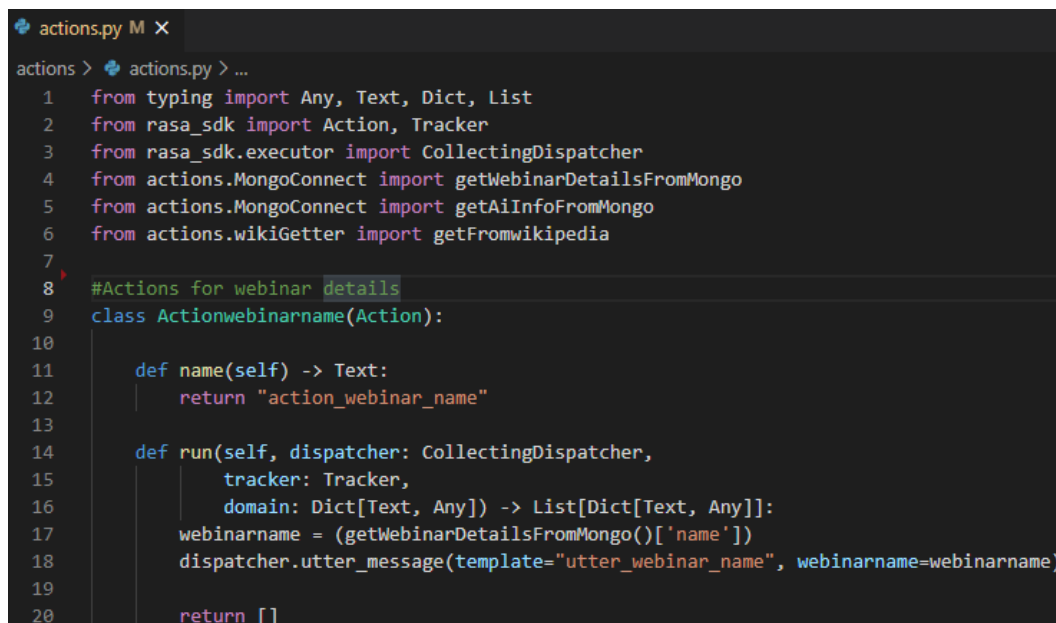


Figure 3.8 : les réponses de chatbot

Comme nous l'avons dit au chapitre 3, les actions personnalisées sont définies dans un fichier appelé actions.py. Ayons un coup d'œil à notre fichier d'actions de chatbot (figure 3.11).



```
actions.py M X
actions > actions.py > ...
1 from typing import Any, Text, Dict, List
2 from rasa_sdk import Action, Tracker
3 from rasa_sdk.executor import CollectingDispatcher
4 from actions.MongoConnect import getWebinarDetailsFromMongo
5 from actions.MongoConnect import getAiInfoFromMongo
6 from actions.wikiGetter import getFromwikipedia
7
8 #Actions for webinar details
9 class Actionwebinarname(Action):
10
11     def name(self) -> Text:
12         return "action_webinar_name"
13
14     def run(self, dispatcher: CollectingDispatcher,
15             tracker: Tracker,
16             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
17         webinarname = (getWebinarDetailsFromMongo()['name'])
18         dispatcher.utter_message(template="utter_webinar_name", webinarname=webinarname)
19
20     return []
```

Figure 3.9 : Rasa core : fichier des actions

L'instruction au-dessus de l'importation importe des modules (tels que rasa sdk), ce qui est extrêmement nécessaire pour confirmer que le serveur d'action personnalisé et le serveur exécutant le bot peuvent échanger des informations.

Après cela, nous définissons la classe pour l'action personnalisée. Il contient deux fonctions run et name, qui doivent correspondre aux noms des actions personnalisées dans les récits de formation. Par exemple, lorsque l'action personnalisée action\_hello\_world est incluse dans une histoire, Rasa sait exécuter le code défini dans la classe d'action personnalisée nommée action\_hello\_world.

La fonction run dans la classe contient le code que nous voulons exécuter, une fois que l'action personnalisée est attendue. La fonction d'exécution est l'endroit où nous pouvons spécifier ce que fait réellement l'action personnalisée. Notez les éléments tracker et dispatcher, qui sont des parties très utiles et importantes de la fonction d'exécution :

- **Tracker** : garde une trace de la progression d'un dialogue à chaque étape : quelles entités ont été extraites, quelles intentions ont été prédites, ainsi que d'autres informations
- **Dispatcher** : La réponse est renvoyée à l'utilisateur par ce composant.

La mise à jour de ce fichier d'actions pour notre chatbot ressemblerait à ceci (figure 4.11). Nous avons créé une classe appelée "Actionwebinarname" qui, lorsque l'action appelée Actionwebinarname est prédite, l'assistant interroge la réponse de la base de données en utilisant

une autre classe qui permet à l'application de se connecter au serveur MongoDB et de renvoyer la réponse à l'utilisateur.

Dans d'autres cas, le chatbot peut ne pas trouver la réponse dans la base de données, nous devons donc traiter ces cas, la meilleure façon de satisfaire les demandes de l'utilisateur, nous allons rechercher les informations sur internet et plus précisément sur Wikipédia.

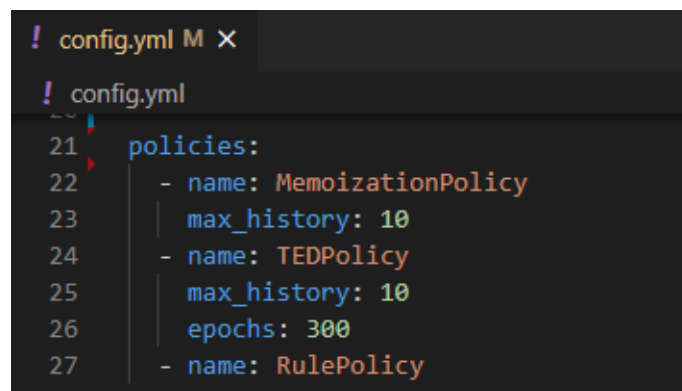
### Bibliothèques Wikipédia et Pymongo :

- **Wikipédia** : a été développé en tant que bibliothèque Python pour accéder aux données de Wikipédia et les analyser. [19]
- **PyMongo** : Il propose des outils permettant d'interagir avec MongoDB depuis Python. [20]

### 3.4.3 Politiques de dialogue

Composant de politiques qui forme le modèle de dialogue et joue un rôle très important dans la définition de son comportement. Certaines de ces politiques sont très simples, comme celles qui reflètent les conversations sur lesquelles ils ont été formés, mais certaines d'entre elles sont assez complexes, du genre qui s'appuient sur l'apprentissage automatique pour prédire la prochaine étape en fonction du contexte de la discussion.

Semblable au pipeline NLU que nous avons configuré précédemment, les politiques de dialogue sont également configurées dans le fichier config.yml, que vous trouverez dans le répertoire principal du projet. Examinons la configuration par défaut (Figure 3.13) :



```
! config.yml M X
! config.yml
21 policies:
22   - name: MemoizationPolicy
23     max_history: 10
24   - name: TEDPolicy
25     max_history: 10
26     epochs: 300
27   - name: RulePolicy
```

Figure 3.10 : Rasa core : Policies

- Une politique de dialogue simple que nous allons examiner est la politique de mémorisation (Memoization Policy). Il coupe la partie de la conversation définie par max\_history (donc si max\_history : 10, 10 revient en arrière) et recherche un fragment d'histoire correspondant dans l'ensemble de données d'apprentissage. Il prédit la même action actuelle à partir des données d'entraînement s'il trouve une correspondance.
- Politique de dialogue d'intégration du transformateur (Transformer Embedding Dialogue (TED) Policy) :
  - **Description** : Nous présentons ici une politique de dialogue basée sur une architecture de transformateur dans laquelle le mécanisme d'auto-attention fonctionne tout au long de la séquence des cycles de dialogue.

Des travaux récents ont utilisé des réseaux de neurones répétitifs hiérarchiques pour coder plusieurs déclarations dans le contexte du dialogue, mais nous pensons que le mécanisme d'attention pure est plus approprié. Par défaut, un réseau neuronal récurrent (Recurrent Neural Network : RNN) suppose que chaque élément d'une séquence est pertinent pour créer un codage de la séquence complète, mais une seule conversation peut consister en plusieurs segments de parole qui se chevauchent tandis que les locuteurs intercalent plusieurs sujets. Un transformateur choisit qu'il transforme pour inclure dans son codage l'état du dialogue en cours et est naturellement adapté pour ignorer sélectivement ou prêter attention à l'historique du dialogue.

#### **- Étapes d'architecture :**

1. Pour chaque pas de temps, rassemblez toutes les entrées de l'utilisateur (intention de l'utilisateur et entités), les actions système précédentes, les emplacements et les formulaires actifs, puis combinez-les dans un vecteur d'entrée pour l'intégration du pré-transformateur.
  2. Alimentez-le au transformateur.
  3. Pour chaque pas de temps et afin d'obtenir des plongements d'un dialogue, nous devons appliquer une couche dense à la sortie du transformateur.
  4. Une couche dense est appliquée à chaque pas de temps pour créer des incorporations pour les actions du système.
  5. Découvrez (par calcul) si l'intégration du dialogue et les actions du système intégré sont similaires. Cette étape est basée sur le StarSpace qui est un modèle d'intégration neuronale à usage général qui peut résoudre une grande variété de problèmes.
- Politique de règle : est une politique qui traite les parties de conversation qui suivent un comportement fixe (par exemple, la logique métier). Une prédiction est faite à l'aide des règles que nous avons stockées dans nos données d'entraînement.

#### **3.4.4 Échouer gracieusement à Rasa**

Une autre amélioration précieuse que nous pouvons apporter à notre assistant consiste à mettre en œuvre une politique de repli. La politique de secours permet de s'assurer que si notre assistant fait une erreur, il gère la situation normalement.

Bien que Rasa généralise aux messages invisibles (message que nous n'avons pas dans le fichier NLU de formation), certains messages peuvent recevoir une faible confiance de classification. L'utilisation de secours aidera à garantir que ces messages à faible niveau de confiance sont traités avec élégance, en donnant à l'assistant la possibilité de répondre avec un message par défaut ou de tenter de dissiper l'entrée de l'utilisateur.

#### **3.4.5 Former et tester l'assistant**

Une fois les actions personnalisées et le domaine en place, nous pouvons former la première version de notre chatbot et voir comment cela fonctionne. Et pour faire ce travail, nous devons entrer la commande "rasa train" dans la cli. Une fois le modèle formé, nous pouvons le tester en ligne de commande avec la commande "rasa shell" (figure 3.12) et nous devons démarrer le serveur d'actions personnalisées à l'aide de la fonction Rasa CLI "rasa run actions".

Les actions personnalisées s'exécutent sur un serveur distinct de celui sur lequel les modèles s'exécutent. Rasa appellera un point de terminaison, que vous spécifiez, lorsqu'une action personnalisée est prédite. Ce point de terminaison doit être un serveur Web séparé qui répond

à l'appel, exécute le code et, éventuellement, renvoie des messages pour étendre l'état du dialogue. Nous pouvons trouver tous les détails sur le serveur d'action personnalisé dans le fichier de projet. Répertoire nommé endpoints.yml dans lequel nous devons configurer le serveur et le port d'écoute :

action\_endpoint :

URL : « <http://localhost:5055/webhook> »

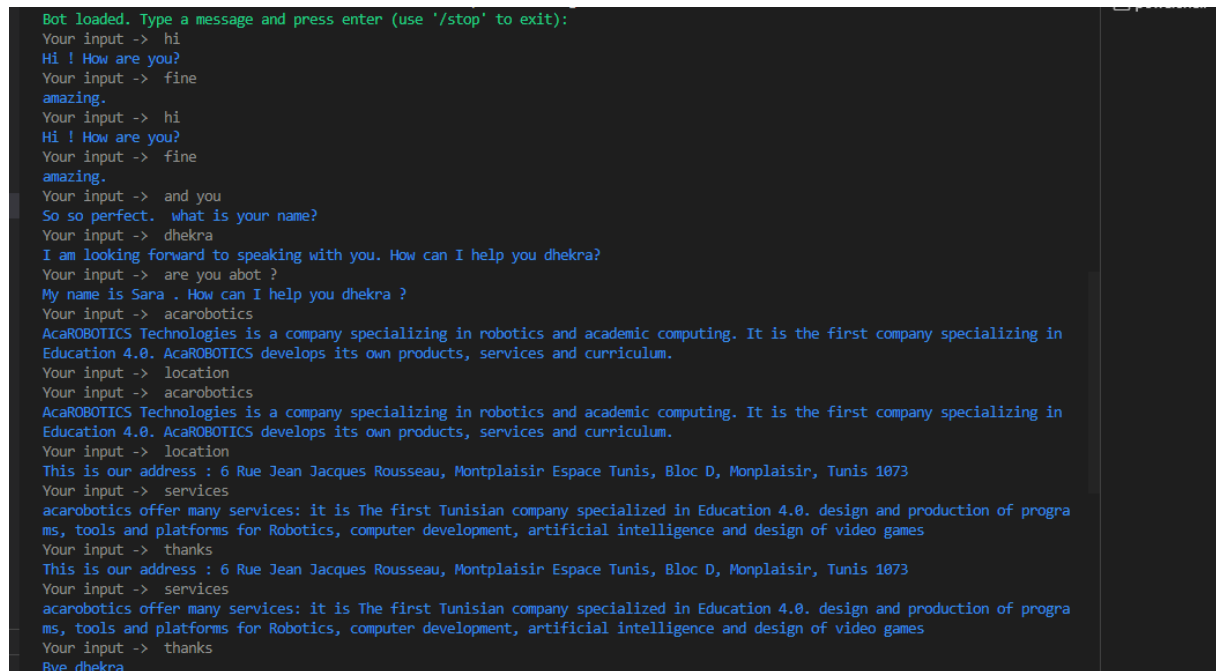
A screenshot of a terminal window with a dark background and light-colored text. The text shows a series of interactions between a user and a chatbot. The chatbot's responses are consistently formatted, starting with a greeting or acknowledgment, followed by a question or statement, and then a detailed response. The conversation includes topics like the chatbot's name (Sara), its company (AcaROBOTICS), its location (6 Rue Jean Jacques Rousseau, Montplaisir Espace Tunis, Bloc D, Monplaisir, Tunis 1073), and its services (design and production of programs, tools and platforms for Robotics, computer development, artificial intelligence and design of video games). The chatbot also includes a closing statement 'Bye dhekra'.

Figure 3.11 : test

## 3.5 Base de données

### 3.5.1 MongoDB

Les développeurs décrivent MongoDB comme "La base de données des idées géantes". MongoDB stocke des données comme des documents dont la structure peut varier, offrant un schéma dynamique et flexible. MongoDB a également été conçu pour une haute disponibilité et une évolutivité, avec une réplication intégrée et un partitionnement automatique. Et c'est exactement ce que nous voulons dans notre projet.

### 3.5.2 Traitement des données

Lorsque les données sont collectées et converties en informations utiles, le traitement des données a lieu.

Généralement effectué par une équipe de data scientists ou un seul data scientist. Afin d'assurer la qualité de la sortie des données, il est essentiel de traiter les données correctement.[21]

## Base de données

Dans un monde unique, un "ensemble de données" est un rassemblement de données. En d'autres termes, un ensemble de données correspond au contenu d'un tableau de base de données ou d'une matrice statistique, où chaque colonne du tableau représente une variable spécifique

et chaque ligne correspond à un membre spécifié de l'ensemble de données en question. Ce projet nous oblige à rassembler deux ensembles de données. L'un contient des détails sur le webinaire tels que le nom du webinaire, l'instructeur, la date, le coût, etc. La seconde contient des informations sur le domaine de l'intelligence artificielle, c'est-à-dire que son glossaire et d'autres colonnes contiennent d'autres informations.

Afin de collecter des données, nous utiliserons deux méthodes :

1. **Web Scraping** : Web Scripting est une méthode automatique pour obtenir de grandes quantités de données à partir de sites Web. La majorité de ces données sont au format HTML et sont ensuite converties en format structuré dans un tableur ou une base de données, ce qui permet de les utiliser dans une variété de programmes. [22]
2. **Manual Scraping (Grattage manuel)** : le processus consiste à copier et coller du contenu Web, ce qui est très répétitif et demande beaucoup d'efforts. Parce que nos besoins ne concernent pas seulement grand ensemble de données. En effet, nous ne répondons rien à l'utilisateur. Nous voulons répondre à l'utilisateur avec des données soigneusement conservées.

### 3.6 Conclusion

Ce chapitre explique comment sélectionner le bon pipeline et tester ses performances, ainsi que comment développer le noyau de l'assistant avec ses différents éléments, et même parler avec le chatbot. Nous avons également discuté de la base de données à utiliser.

# Chapitre 4

## 4 Realisation de Partie 2 : Implémentation and Déploiement

### 4.1 Introduction :

Dans ce chapitre, nous allons nous intéresser au déploiement et à la mise en place du chatbot, la première partie contiendra l'hébergement du chatbot sur un serveur public, puis nous intégrerons le chatbot dans un site Web de webinaire.

### 4.2 Rasa X

#### 4.2.1 Obtenir de bonnes données d'entraînement :

Jusqu'à présent, nous avons créé des données de formation pour l'assistant de webinaires en imaginant les choses que les utilisateurs pourraient dire et les écrire comme exemples de formation.

Bien que ce soit une bonne façon de commencer, il y a quelques limitations majeures. L'ensemble de données que nous avons généré (fichier NLU) est très petit et il est intrinsèquement biaisé - nos vrais utilisateurs sont susceptibles de dire des choses à l'assistant que nous n'avons pas anticipées. A cause de cela, l'assistant aura du mal à généraliser. Bien qu'il soit capable de gérer des énoncés et des chemins d'histoire qui sont très similaires à ce que nous l'avons entraîné à gérer, il pourrait trébucher si présenter avec un message inattendu. Les meilleures données de formation que nous pourrions utiliser seraient un grand ensemble de données composé de conversations réelles. De nombreux ensembles de données conversationnelles publiques sont disponibles, mais la plupart d'entre eux sont destinés à usage général : ce ne sont pas des conversations avec nos utilisateurs, et ils ne concernent pas notre domaine.

Rasa X résout ce problème en nous aidant à créer le jeu de données idéal : un grand nombre des conversations qui ont eu lieu entre nos utilisateurs et l'assistant nous permettent de partager l'assistant avec les vrais testeurs afin de recueillir leurs conversations, et plus tard, quand le chatbot a été implémenté, collectez les conversations de vrais utilisateurs. Rasa X convertit ces conversations en données de haute qualité qui peuvent être utilisées pour recycler et améliorer l'assistant

#### 4.2.2 Déploiement de Rasa X

Cet outil d'interface utilisateur est conçu spécifiquement pour aider les développeurs à créer des assistants avec Rasa Open source. Il résout deux problèmes :

1. Tout d'abord, pour simplifier l'utilisation de vrais chats en tant que données de formation.
2. Deuxièmement, pour nous permettre, en tant que propriétaire de chatbot, de revoir les conversations passées à la recherche de modèles ou les erreurs.

La méthode la plus précise pour exécuter Rasa X consiste à le déployer sur un serveur. L'utilisation locale est possible, mais pour vraiment commencer à recueillir des conversations avec les utilisateurs, Rasa X doit être disponible sur Internet à tout moment.

Il est possible de déployer Rasa X sur des services d'hébergement comme Azure, Digital Ocean, AWS, etc. nous voudrions simplement nous assurer que notre machine répond aux exigences minimales :

- Au moins 4 Go de RAM
- 2 à 6 processeurs virtuels
- 100 Go d'espace disque
- Distribution Linux pouvant exécuter Docker (par exemple, Debian 9, Ubuntu 16.04 / 18.04)

## Configurer l'instance de machine virtuelle

Dans ce projet, nous utiliserons Azure pour les étudiants. Nous nous connecterons à notre compte azur et accédez aux services Azure puis Créer une ressource et Créer une machine virtuelle (Figure 4.1)

Figure 4.1 : créer une machine virtuelle

Après avoir choisi les bonnes options pour notre VM. Nous devons configurer une adresse IP statique pour la machine (et une fois que c'est fait, la machine virtuelle est prête.

## Installer Rasa X

Nous avons créé notre machine virtuelle, mais jusqu'à présent, elle est vide. Connectons-nous à la nouvelle instance utilisant SSH (Figure 4.2), afin que nous puissions télécharger et installer Rasa

```
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NeverMind>echo off
ssh -i rasaX@52.254.67.172
rasaX@52.254.67.172's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-1111-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-1111-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

13 packages can be updated.
5 of these updates are security updates.
To see these additional updates run: apt list --upgradable
```

Figure 4.2 : connexion SSH

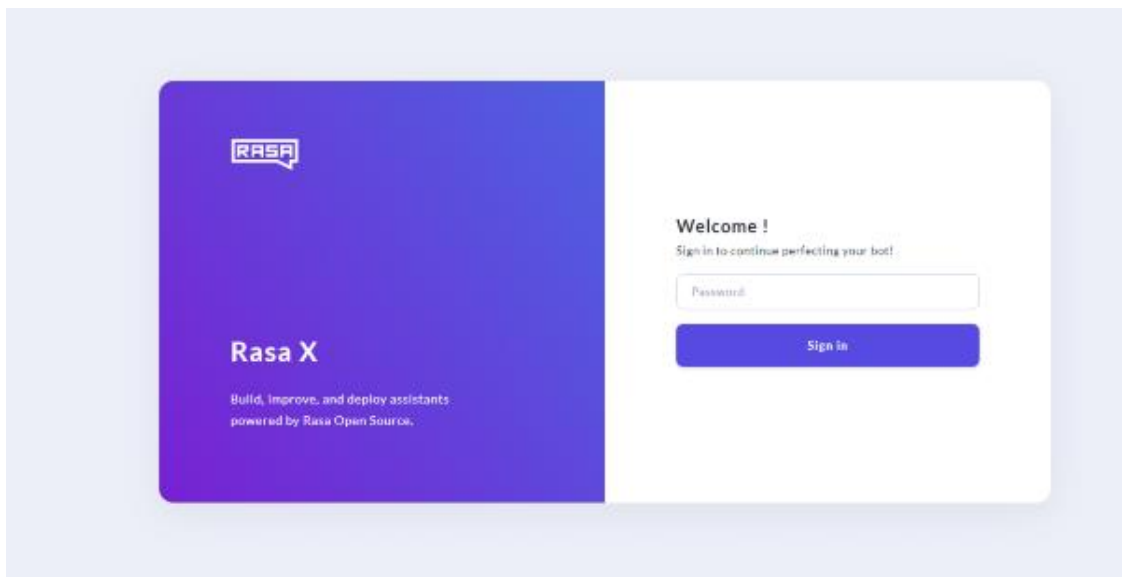


Maintenant, nous devons télécharger le script d'installation et l'installer avec cette commande :

- `curl -sSL -o install.sh https://storage.googleapis.com/rasa-x-releases/0.39.3/install.sh`
- `sudo bash ./install.sh`

Démarrez Rasa X avec cette commande [`sudo docker-compose up -d`] et attendez que tous les conteneurs soient en cours d'exécution.

Pour accéder à Rasa X, il suffit de définir le mot de passe administrateur et Rasa X est prêt (figure 4.3)



**Figure 4.3 : Rasa X**

## **Connecter notre chatbot à Rasa X**

Ici, nous devons lier l'instance Rasa X à un référentiel git distant, comme Gitlab ou GitHub pour suivre l'historique des versions des données d'entraînement. En plus d'obtenir un enregistrement de toutes les modifications. Grâce au contrôle de version intégré, les équipes peuvent profiter des avantages en aval d'un workflow de développement basé sur Git : examen des modifications avant ils entrent en production, effectuent des tests sur les modifications proposées et intègrent CI/CD.

### **Structure du projet**

Pour que le contrôle de version intégré se synchronise avec le référentiel distant, la structure du fichier doit suivre la même structure générée lorsque nous créons un nouveau projet à l'aide de "rasa init".

Pour configurer cela, nous devons d'abord télécharger nos fichiers de projet sur un référentiel GitHub distant (Figure 4.4)

|                                       |                     |             |
|---------------------------------------|---------------------|-------------|
| ./vscode                              | use mongo as db     | last month  |
| actions                               | after stories error | 29 days ago |
| data                                  | after meeting       | 17 days ago |
| models                                | before meeting v3   | 18 days ago |
| tests                                 | before deploying    | last month  |
| README.md                             | use mongo as db     | last month  |
| Voicebot.py                           | before deploying    | last month  |
| XBOT.ipynb                            | before deploying    | last month  |
| XBot_architecture_of_ai_component.txt | before deploying    | last month  |
| XBot_definition_of_ai_component.txt   | before deploying    | last month  |

Figure 4.4 : GitHub

Maintenant, nous devons activer le contrôle de version intégré dans le tableau de bord Rasa X. Ensuite, générer des clés SSH pour établir une connexion sécurisée entre Rasa X et GitHub, et enfin connectez-vous au référentiel dans Rasa X (Figure 4.5).

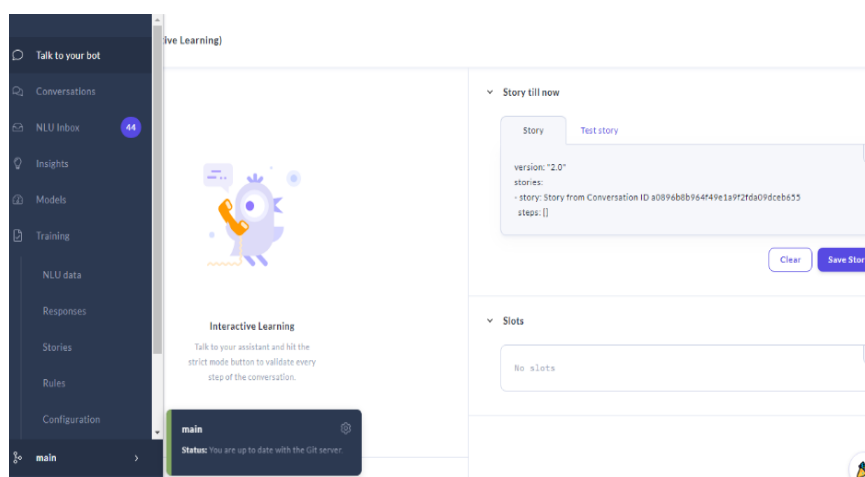


Figure 4.5 : Rasa X

## Configurer le serveur d'actions

Nous avons une autre chose à configurer : le serveur d'actions personnalisées de l'assistant. Pour ce faire, nous allons placer le code d'action personnalisé de l'assistant dans un répertoire d'actions sur le serveur.

Nous aimerions maintenant former un fichier `docker-compose.override.yml`. Ce fichier indique à docker-compose pour lancer un serveur d'action personnalisé une fois le serveur Rasa X démarré.

## Docker hub

Ici, nous utilisons deux images, la première est une image de serveur d'actions et la seconde est une image MongoDB et les deux sont construits et poussés vers un référentiel cloud (Docker Hub) :

- **Image du serveur d'actions** : consiste à exécuter nos actions personnalisées, il s'agit du [DockerFile](#) (Figure 4.6) qui nous permettent de construire cette image.

```

rasaX@RasaX: /etc/rasa$ cat Dockerfile
# Extend the official Rasa SDK image
FROM rasa/rasa-sdk:2.4.1

# Use subdirectory as working directory
WORKDIR /app

# Copy any additional custom requirements, if necessary (uncomment next line)
COPY actions/requirements-actions.txt ./

# Change back to root user to install dependencies
USER root

# Install extra requirements for actions code, if necessary (uncomment next line)
RUN pip install -r requirements-actions.txt

# Copy actions folder to working directory
COPY ./actions /app/actions

# By best practices, don't run the code with root user
USER 1001
rasaX@RasaX: /etc/rasa$

```

Figure 4.6 : fichier docker

- **Image MongoDB** : Cette image sera la base de données. Il contiendra tous nos jeux de données et sera connecté aux autres images. Ceci est sur les images officielles de docker, nous n'avons donc pas besoin de le construire à partir de zéro.

Après avoir connecté toutes les images dans le même réseau et s'être assuré qu'elles sont connectées l'un à l'autre. Nous pouvons redémarrer le conteneur Docker Rasa X et l'assistant sera entièrement fonctionnel sur Rasa X. Nous allons maintenant parler à notre chatbot (Figure 4.7) pour nous assurer qu'il est fonctionné correctement.

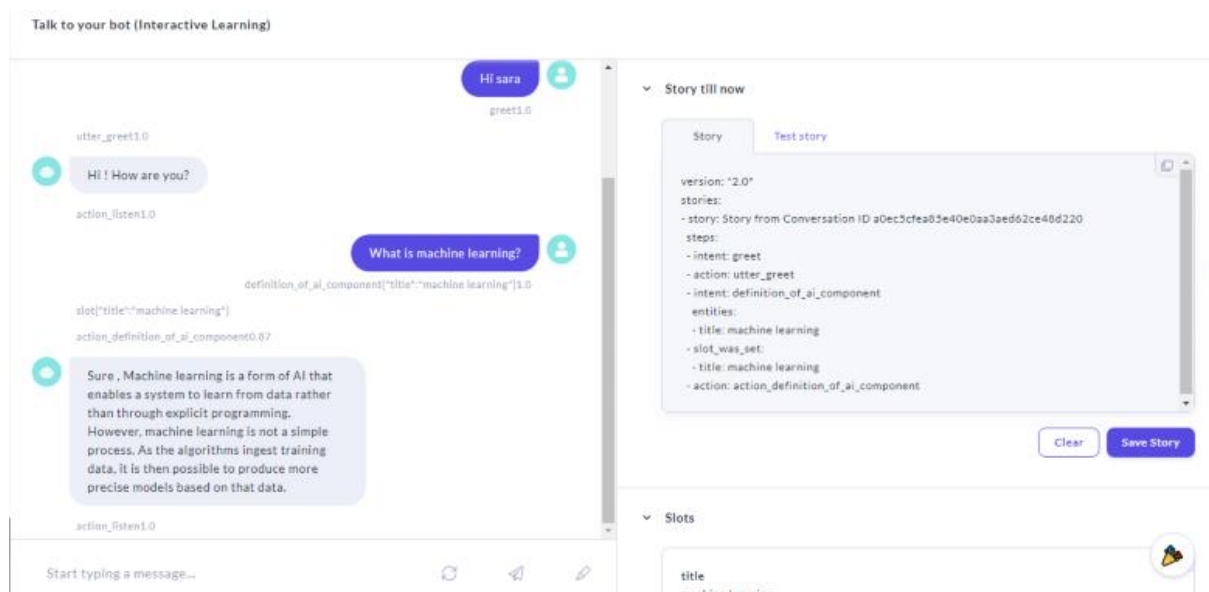


Figure 4.7 : parler avec le chatbot

Une fois que le chatbot est capable de gérer les histoires de chemin heureux les plus importantes, l'étape suivante consiste à inviter des testeurs - de vrais utilisateurs - à tester et à discuter avec l'assistant.

En utilisant les conversations que nous recueillons lors des tests, nous pourrions examiner les conversations et apporter des modifications mineures, majeures et architecturales pour améliorer le chatbot.

Rasa X nous permet d'inviter de vrais testeurs à parler à notre assistant via un lien partageable. Pour générer ce lien, nous devons d'abord saisir des informations sur l'assistant .

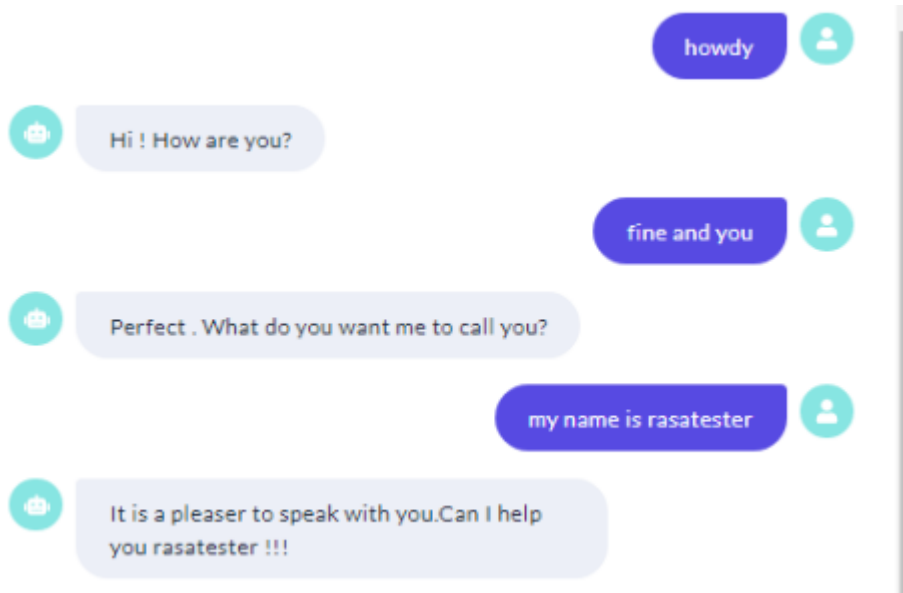
### **Partage avec les testeurs :**

Une fois qu'un utilisateur de test invité clique sur le lien que nous avons partagé, il verra une interface utilisateur de chat simple (Figure 4.9) où ils peuvent immédiatement parler à votre assistant. Il n'y a pas de logiciel pour installer ou configurer pour le testeur invité, afin qu'il puisse immédiatement commencer les tests.

### **Examen des conversations de test :**

Une fois que nos testeurs invités ont eu l'occasion de discuter avec l'assistant, nous pouvons examiner leurs conversations dans Rasa X.

L'examen de ces conversations collectées devrait être une excellente source d'apprentissage sur l'assistant et aider à décider quels changements sont nécessaires pour améliorer Sara.



**Figure 4.8 : tester la conversation**

### 4.3 Connexion aux canaux de messagerie

Une fois que nous estimons que l'assistant peut gérer les conversations avec les testeurs invités, il est temps de passer à l'étape suivante consistant à se connecter à des canaux de messagerie externes.

Une plate-forme de messagerie externe peut être l'une des nombreuses plates-formes : Slack, Facebook Messenger, un site Web ou toute autre méthode où de vrais utilisateurs parleront au chatbot. Nous recueillerons les conversations des utilisateurs avec l'assistant et continuera à apporter des améliorations basées sur ce que nous pouvons apprendre de ces conversations du monde réel.

#### 4.3.1 Meilleur logiciel de webinaire sur le marché

La popularité croissante des webinaires a entraîné une multitude de nouvelles plates-formes, je vais mettre en évidence quelques-uns des plus populaires d'entre eux. [23]

- **WebinarJam** : Un type d'abonnement payant et est le meilleur actuellement disponible. Une plateforme facile à utiliser pour les personnes qui ont un savoir-faire technique limité dans la mise en place.

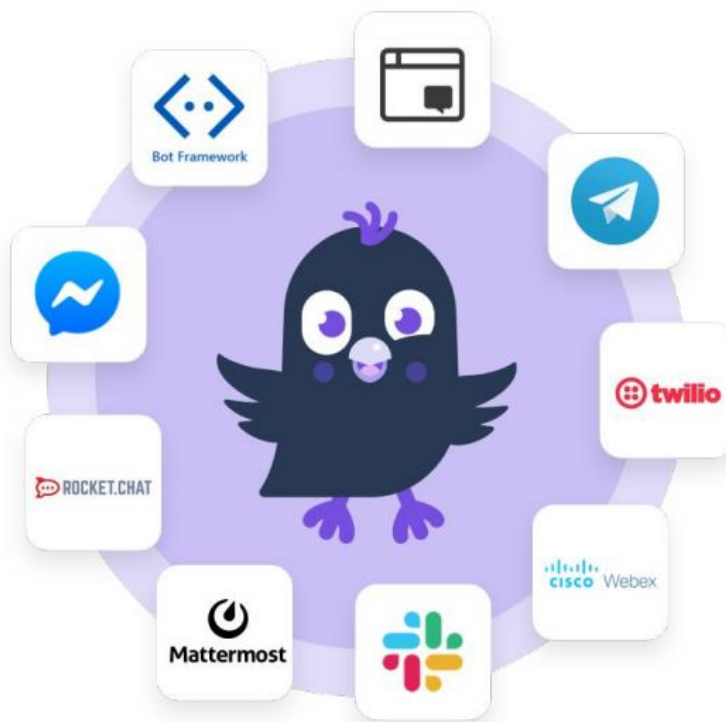


Figure 4.9 : canaux de messagerie

Un webinaire capable d'accueillir automatiquement un large public.

- **Zoom** : logiciel de webinaire entièrement basé sur le cloud. L'un des meilleurs logiciels du marché qui effectue un large éventail d'activités, ce qui en fait un atout majeur pour l'entreprise. Offre également des options de vidéo et audio HD, de partage d'écran, de bureau et de partage d'applications.

Il existe de nombreuses autres plateformes de webinaires mais leurs principaux problèmes sont : d'entre eux sont payants, d'autre part, cela nécessite parfois un long processus de téléchargement ou inscription et ainsi de suite.

#### **4.4 Conclusion**

Le but de ce chapitre est de présenter les interfaces les plus importantes du projet, tout en justifiant nos choix technologiques et en expliquant les différentes phases d'utilisation.

## **Conclusion générale :**

Le travail réalisé lors de ce projet de fin d'étude a porté sur le développement et le déploiement d'un assistant personnel afin de permettre aux participants du webinaire à distance de poser leurs questions et de parler avec un chatbot intelligent.

Le projet a été mis en œuvre en plusieurs étapes. Une première étape consiste à comprendre le contexte général du projet. Une seconde dans le but d'analyser et de faire émerger les besoins réels de l'utilisateur. Cette étape est l'une des étapes clés de la réussite de tout projet. C'est pourquoi j'ai passé beaucoup de temps sur l'analyse et la spécification des besoins.

La troisième étape consiste à se familiariser avec Rasa et à apprendre à l'utiliser, et une fois le chatbot est prêt, nous allons mettre en place une plateforme de webinaires organisés à distance où l'assistant sera mis en place. Nous avons réussi à développer la fonctionnalité de l'application au fil du temps. Le test de ce dernier s'est déroulé avec succès ce qui signifie que l'application est maintenant prête à être hébergée. Bien que nous ayons rencontré quelques difficultés au début du projet, c'était une bonne occasion de sortir du cadre théorique et d'appliquer les connaissances acquises durant les études universitaires dans le monde du travail. De plus, ce projet nous a permis de mettre en pratique les connaissances acquises sur le cycle de vie d'un projet informatique.

Cette réalisation a également été l'occasion d'intégrer le métier monde et apprendre plusieurs compétences et habitudes problèmes sociaux tels que le travail de groupe et la collecte d'informations, ceci afin d'extraire les besoins du système acteurs à mettre en œuvre. Ce projet est un noyau sur lequel plusieurs perspectives peuvent s'étendre pour l'enrichir et le développer. En effet, le travail que nous avons réalisé est modulaire et évolutif.

Il est donc possible de :

Améliorez le projet en permettant à l'assistant de converser en différentes langues. Nous pouvons également inclure le chatbot sur plusieurs plateformes.

## Références

- [1] <https://digitalhumans.com/>.
- [2] <https://developers.google.com/hangouts/chat/concepts/bots>.
- [3] <https://www.futura-sciences.com/tech/definitions/internet-chatbot-15778>.
- [4] <https://www.comparethecloud.net/articles/chatbot-revolution-evolution-ai>.
- [5] <https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next>.
- [6] <https://rasa.com/docs/rasa>.
- [7] <https://rasa.com/docs/rasa-x>.
- [8] <https://opensource.com/resources/what-docker>.
- [9] <https://www.mongodb.com/what-is-mongodb>.
- [10] <https://code.visualstudio.com>.
- [11] <https://www.python.org/doc/essays/blurb>.
- [12] <https://angular.io/docs>.
- [13] <https://azure.microsoft.com/en-us>.
- [14] <https://hub.docker.com>.
- [15] <https://github.com>.
- [16] <https://www.heroku.com>.
- [17] <https://monkeylearn.com/blog/natural-language-understanding>.
- [18] <https://blog.rasa.com/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture>
- [19] <https://pypi.org/project/wikipedia>.
- [20] <https://pypi.org/project/pymongo>.
- [21] <https://www.sisense.com/glossary/data-preparation>.
- [22] <https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it>.
- [23] <https://www.ventureharbour.com/webinar-software-10-best-webinar-platforms-compared>.



## Résumé

Ce rapport présente le travail de quatre mois d'un stage à AcaROBOTICS Technologies dans le cadre d'obtention du Diplôme National de génie mécatronique

Ce projet vise à créer un assistant personnel utilisant le framework rasa pour assister les utilisateurs lors d'un webinaire tenu à distance, puis pour implémenter l'assistant dans la plateforme de webinaires

Mots clés : Angular, Python, Tensorflow/Keras, RASA, NLU, RNN, apprentissage en profondeur, MongoDB.