Documentation scripts

Document expliquant le fonctionnement des scripts
Comment en créer et les intégrer au projet

1 Introduction

Ce document explique le fonctionnement des scripts de l'agent conversationnel utilisés dans le cadre du projet "Chat Bot".
Ce document contient également des indications afin d'intégrer de potentielles nouveaux scripts au sein du projet. Cela peut être d'avoir plusieurs scripts si l'on souhaite proposer aux utilisateurs des personnalités différentes.

Note : Le système utilisé pour générer les scripts est assez rudimentaire, la conception d'un outil, avec interface graphique, simplifiant la génération et la modification des scripts a été envisagé pendant un moment, mais par manque de temps (et puisque ce n'est qu'un prototype), nous ne l'avons pas mis au point.

2 Description d'un script

2.1 En général

Chaque script correspond à un fichier JSON contenus dans le répertoire "resource/raw/" et dont le nom correspond au nom du robot associé au script.
Exemple : "rob.json"

Chaque fichier est constitué de trois parties (voir annexe pour un exemple de fichier) : "Robot", "Humain" et "Lien".

- "Robot" : Contient l'ensemble des questions posées par l'agent conversationnel, ainsi qu'un numéro d'identification (entre 0 et 999) et une action (voir section 2.2)
- "Humain" : Contient l'ensemble des réponses possibles par les utilisateurs, ainsi qu'un numéro d'identification (à partir de 1000) et une action (voir section 2.2)
- "Lien" : Permet de faire le lien entre les questions du robot et les réponses possibles de l'utilisateur pour chaque question.

Cette structure a été construite de telle manière à pouvoir représenter un "flow chart" tel que celui présent en annexe 1. La partie "Robot" représente les messages des boîtes rouges, la partie "Humain" représente les messages qui font la transition entre deux boîtes rouges et la partie "Link" est celle qui fait le lien entre les deux.
Par exemple, si l'on souhaite représenter une partie de ce flow chart (les quatre premières boîtes rouges, de haut en bas) avec leurs réponses possibles, le résultat ressemblera à ce qui est contenu dans l'annexe 2.

Cette structure nous permet de pouvoir changer facilement le texte des messages (robot ou humain) sans devoir changer la structure, cela permet également d'éviter d'avoir de la redondance puisqu'un message, même s'il est utilisé à plusieurs endroits dans le flow chart, est déclaré à un seul endroit et seul son numéro d'identification est utilisé.

2.2 Les actions

Certains messages (robot ou humain) sont liés à des actions. Par exemple, l'une des options de l'utilisateur est de pouvoir utiliser la géolocalisation, si l'utilisateur tape sur cette option, il faut que l'on sache que l'on doit utiliser la géolocalisation et cela se fait grâce à un champ "action", optionnel, qui peut être lié à n'importe quel message (robot ou humain).
Les actions liées aux humains sont des actions contextuelles, qui ne se déclenchent que sur les transitions, comme taper du texte ou faire un choix. Alors que les actions liées au robot sont des actions qui se déclencheront toujours lorsqu'on atteindra l'état correspondant (la boîte rouge), comme pour l'affichage des résultats ou la suppression des

notifications.

N'importe quelles actions peuvent être rajoutées tant que celles-ci sont déclarées dans la classe "TypeAction" et manipulé dans la fonction "manageAction" de la classe "MessageVM" (qui gère le système de message).

Un exemple de message avec une action est disponible en annexe 3.

2.3 Gérer plusieurs personnalités

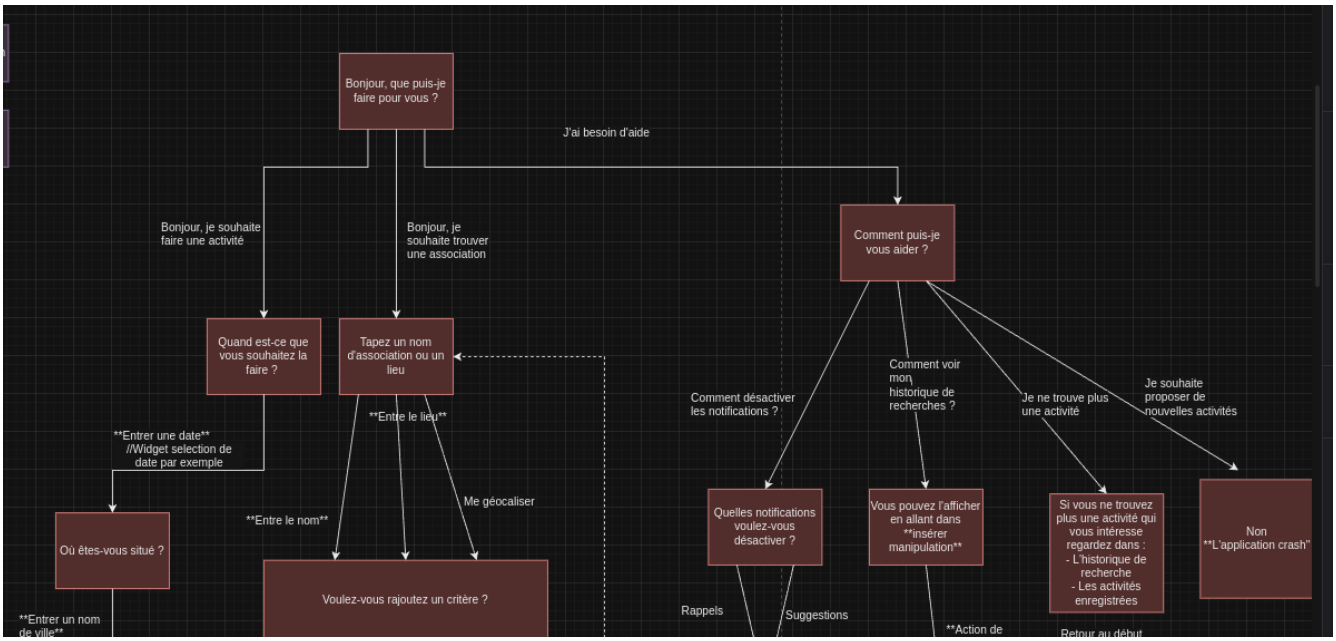
La manière dont le système a été conçu permet d'avoir des scripts correspondant à plusieurs flow chart différents, ils ne sont pas tous obligés de correspondre au même (bien que l'application devra être relancée si l'utilisateur souhaite changer de script en cours de conversation).

Ce système permet également de gérer plusieurs personnalités pour le robot puisqu'il suffit de recréer un fichier (ou de modifier un des fichiers existant) en modifiant les textes pour les faire correspondre à la personnalité du robot, puis de rajouter ce script à la liste des scripts disponible.
Pour ce faire, il suffit de rajouter le script dans la liste "mapScript" de la classe "MessageVM" et de rajouter la possibilité de pouvoir utiliser ce script en rajoutant une option dans la liste "list" de la classe "SettingsComp".

3 Annexes

3.1 FlowChart

Représentation du script grâce à un flowchart

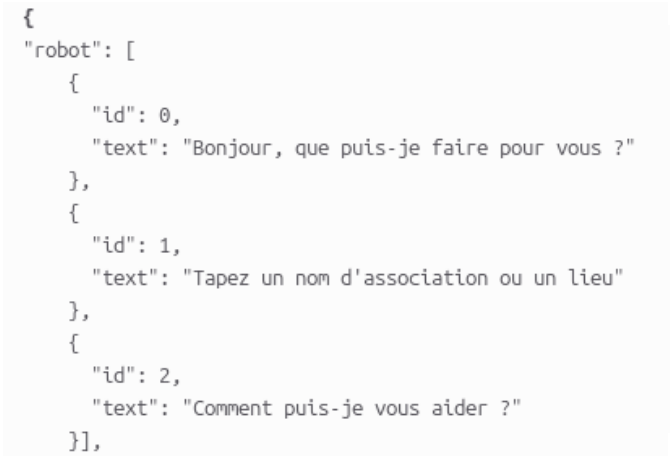


3.2 Correspondance JSON

La correspondance d'une partie du flowchart avec notre structure.

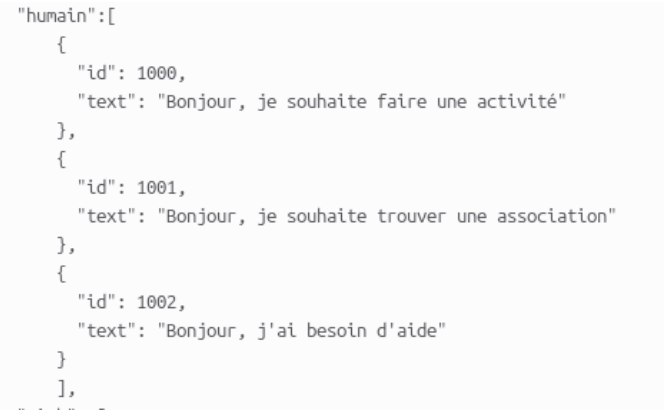
3.3 Avec action

Représentation des actions au sein du code



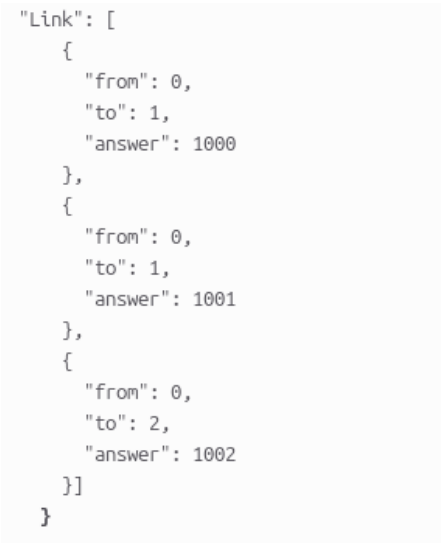
```
{
"robot": [
    {
      "id": 0,
      "text": "Bonjour, que puis-je faire pour vous ?"
    },
    {
      "id": 1,
      "text": "Tapez un nom d'association ou un lieu"
    },
    {
      "id": 2,
      "text": "Comment puis-je vous aider ?"
    }],
```

FIGURE 1 – Partie "Robot"



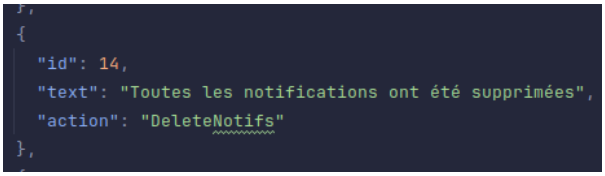
```
"humain":[
    {
      "id": 1000,
      "text": "Bonjour, je souhaite faire une activité"
    },
    {
      "id": 1001,
      "text": "Bonjour, je souhaite trouver une association"
    },
    {
      "id": 1002,
      "text": "Bonjour, j'ai besoin d'aide"
    }
    ],
```

FIGURE 2 – Partie "Humain"



```
"Link": [
    {
      "from": 0,
      "to": 1,
      "answer": 1000
    },
    {
      "from": 0,
      "to": 1,
      "answer": 1001
    },
    {
      "from": 0,
      "to": 2,
      "answer": 1002
    }]
  }
```

FIGURE 3 – Partie "Link"



```
{
  "id": 14,
  "text": "Toutes les notifications ont été supprimées",
  "action": "DeleteNotifs"
},
```

FIGURE 4 – Exemple d'une action dans le fichier json (partie "Robot" car id <= 999)

```
    }

    TypeAction.DeleteNotifs -> {
        disableNotification(context)
    }
```

FIGURE 5 – La même action, mais géré dans la fonction "manageAction" de la classe "MessageVM"

FIGURE 6 – L'action rajoutée dans la classe "TypeAction"