Projet Chatbot / NLP

Rapport technique

Marianne Depierre/Eden Lecarpentier/Jonathan Raso
04/02/22

1. Planning

A. Explication

Afin de réaliser le projet qui a été assigné à notre équipe, nous avons adopté les méthodes Agiles et notamment le framework SCRUM pour pouvoir gérer le temps passé sur chaque tâche.

L'équipe était composée de 3 personnes, avec une cheffe de projet (Marianne Depierre) et 2 développeurs (Eden Lecarpentier et Jonathan Raso). La répartition des différentes tâches a été faite sur un seul sprint, étant donné le délai qui était prévu pour la mise à disposition de notre première version de l'application (5 jours complets).

Les différentes étapes ont été listées par toute l'équipe afin d'avoir de la visibilité et de répartir les tâches entre les 3 membres. Nous avons opté pour une répartition en 3 listes :

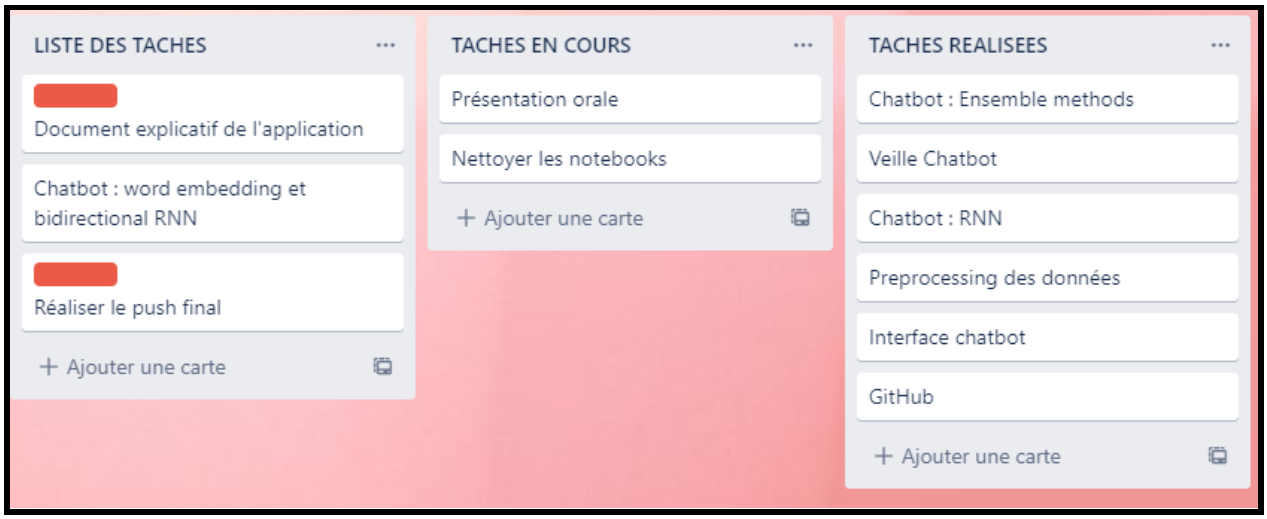
- Liste des tâches
- Tâches en cours
- Tâches réalisées

Vous pourrez voir le détail des différentes étapes dans l'image présente dans la partie B.

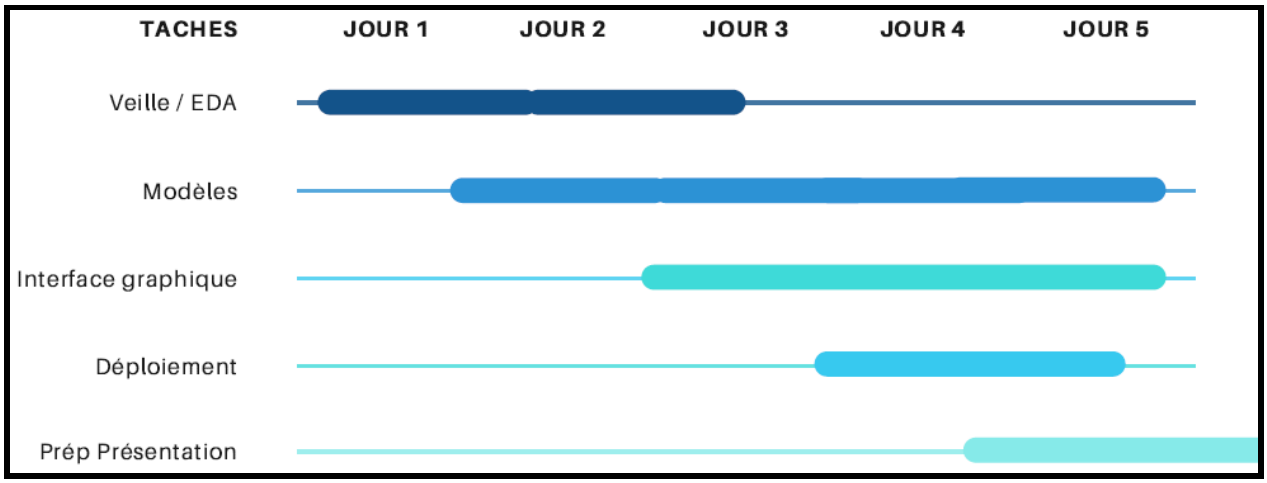
B. Planification via Trello

Le choix de l'outil pour organiser le planning s'est porté sur Trello. Cet outil de gestion de projet en ligne est inspiré par la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.

Vous trouverez ci-dessous le tableau final réalisé sur Trello, qui regroupe l'ensemble des actions réalisées pour la réalisation de notre MVP concernant le projet de chatbot.



Planning disponible sur Trello au 21/01/2021



Planning effectif

2. Description du problème et cas d'usage

A. Problème rencontré

En 2018, Theresa May nomme un ministre de la Solitude, afin de faire des relations humaines une priorité nationale. Ainsi, l'isolement est un problème actuel et non sans conséquences pour les personnes souffrant de solitude.

En effet, une vie sociale réduite agirait sur des maladies chroniques, les pathologies cardiaques, les cancers et impacterait le système immunitaire. Si les personnes âgées semblent être directement concernés, les jeunes le seraient également puisque 40 % des 16-24 ans ont déclaré souffrir de solitude en 2018 en Grande-Bretagne.

L'objectif principal du chatbot ***Maredjo*** est d'assurer une conversation superficielle avec les utilisateurs, afin de tromper la solitude liée à l'isolement des personnes. Il s'agit de créer un outil de discussion en soutien contre l'isolement, capable de répondre aux intéractions courantes en utilisant l'Intelligence Artificielle.

B. Cas d'usage

Dans le cadre de sa démarche "Vivons Heureux", le cabinet de psychiatrie du Dr. Duerf souhaite mettre à disposition de ses patients un chatbot capable d'interagir avec le patient pour des échanges quotidiens.

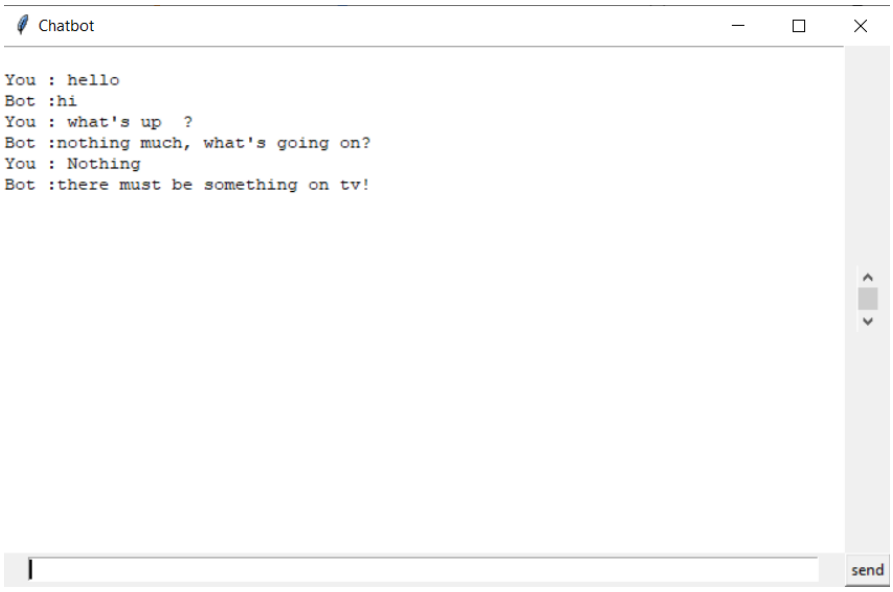
Ces échanges se voulant positifs, les réponses du chatbot sont optimistes. Des blagues et des ragots sont également proposés, à des fins humoristiques ou de divertissement. La patientèle visée par ce projet étant composée à 61% de jeunes âgés de 18 à 25 ans, le design et l'ergonomie du chatbot doivent être attractifs pour cette cible. La solution existante du Chatbot ***Maredjo*** est adaptée afin de mieux correspondre aux objectifs du projet.

3. Cahier des charges

A. Charte graphique

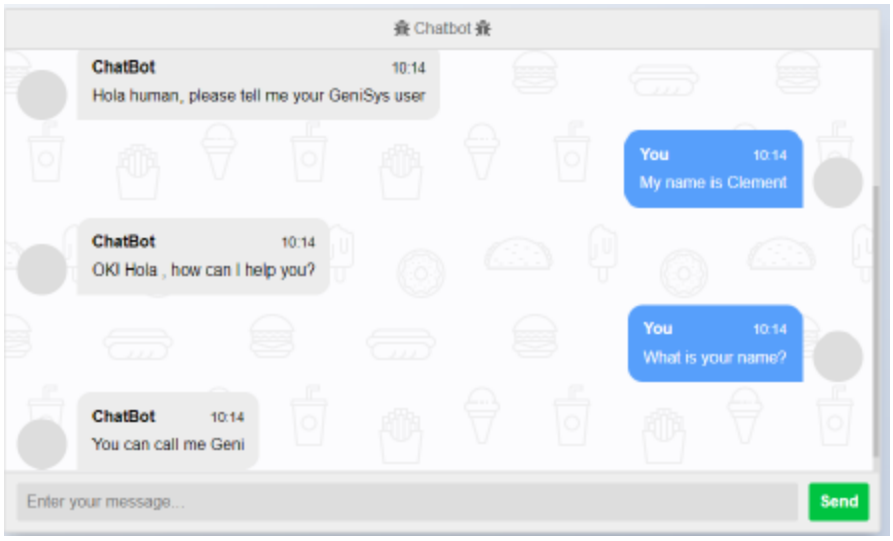
Il n'y a pas de charte graphique précisément définie pour chacun des projets.

Pour la solution applicative, la police utilisée est Courier New, sans-serif. Les couleurs présentes sont le blanc (#FFFFFF) et un gris (#CDCDCD).



Interface graphique de la solution applicative

Pour l'application web la police de texte utilisée est Helvetica, sans-serif. Il y a trois couleurs principales : un blanc pour le fond (#F5F7FA), un bleu (#579FFB) et un vert clair (#00C441).



Interface graphique de l'application web ***Maredjo***

Des avatars peuvent être implémentés à la demande du client à l'emplacement des ronds gris juxtaposés aux messages.

B. Spécificités fonctionnelles

Comme définit ci-dessus le chatbot doit dialoguer avec l'utilisateur pour des échanges de la vie courante et doit proposer des blagues et des ragots à la demande du patient.

Deux solutions ont été proposées. La première est une solution applicative réalisée avec le module Tkinter. Suite à un échange avec le client, une application web a été développée pour mieux concorder avec les objectifs du projet.

- Solution applicative : choix de Tkinter

Pour réaliser l'interface du chatbot, le choix s'est porté sur Tkinter ou "Tk interface". En effet, ce module est intégré nativement dans python et propose les widgets (ou éléments graphiques) nécessaires à l'élaboration du chatbot : une barre de saisie pour que l'utilisateur entre ses questions pour le chatbot, un bouton pour envoyer ces questions au chatbot, une fenêtre qui affiche la conversation de l'utilisateur avec le chatbot et une barre de défilement qui permet de remonter le fil de la discussion.

- Application web : choix de Flask & Heroku

Suite à l'entretien avec le client, le choix de l'interface s'est porté sur Flask pour l'application web. Flask est un microframework et fonctionne donc avec un noyau et des modules annexes. Il s'agit donc d'une solution qui offre des possibilités de personnalisation, tant dans les fonctionnalités que dans le design de l'application. Les éléments graphiques nécessaires décrits ci-dessus pour la solution applicative sont intégrables avec ce choix d'interface. Le déploiement de l'application est réalisé avec Heroku. Il s'agit d'une Plateforme en tant que Service (PaaS). Heroku permet de déployer, gérer et mettre à l'échelle des applications. Il est compatible avec des projets codés en Python et avec Git et GitHub, les outils de versionning utilisés durant ce projet.

- Outils de développement

Langages :

- Python
- HTML
- CSS

Technologies :

- Jupyter Notebook
- Visual Studio Code
- Flask
- Heroku
- Git & GitHub

4. Solutions proposées

Au final, deux solutions sont proposées : une solution logicielle basée sur un modèle de machine learning et une webapp utilisant le deep learning.

A. Solution logicielle (interface tkinter)

- Dataset

	Questions	Answers
0	hi, how are you doing?	i'm fine. how about yourself?
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
2	i'm pretty good. thanks for asking.	no problem. so how have you been?
3	no problem. so how have you been?	i've been great. what about you?
4	i've been great. what about you?	i've been good. i'm in school right now.

Extrait du dataset dialogs.txt utilisé pour la solution applicative

Un premier jeu de données a été fourni par le client au format .txt. Il s'agit d'un recueil de questions-réponses de 3724 enregistrements.

- Préparation des données

La ponctuation est retirée et le texte est converti en minuscules. Chaque mot est ensuite identifié.

Pour chacun de ces mots, on lui associe une fréquence (Term Frequency-Inverse Document Frequency) selon son occurrence dans le document. LTF-ID donne un poids plus important aux termes les moins fréquents, considérés comme plus discriminants. Ce vecteur est utilisé en entrée dans le modèle.

- Modèle

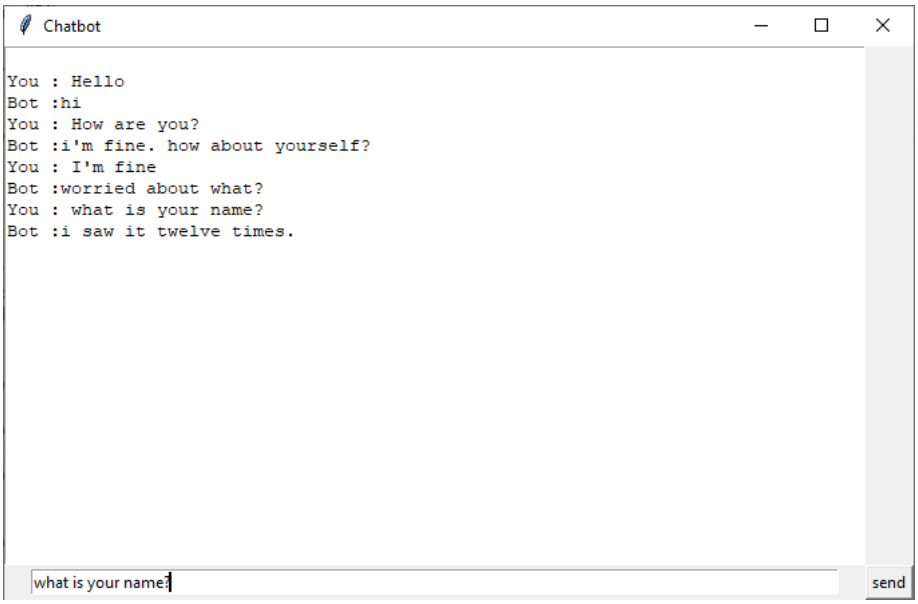
Pour cette première proposition, une solution de machine learning est utilisée. Le modèle retenu est celui du DecisionTreeClassifier ou Classification par arbre de décision.

- Déroulé d'une conversation

Lorsque l'utilisateur exécute l'application, une interface apparaît avec les éléments graphiques (*widgets*) décrits dans le cahier des charges. La fenêtre de conversation représente la majeure partie de l'interface. A droite de cette fenêtre est présente la barre de défilement. La barre de saisie est présente en bas de l'interface, le bouton *send* est à sa droite.

L'utilisateur saisit sa question ou son interaction. Une fois que l'utilisateur a cliqué sur le bouton *send*, son entrée s'efface de la barre de saisie et s'affiche dans la

fenêtre de conversation. Le chatbot répond alors en dessous de la question saisie par l'utilisateur.



L'utilisateur peut enchaîner les questions autant de fois qu'il le souhaite. Lorsque l'utilisateur souhaite arrêter l'exécution du chatbot, il peut cliquer sur la croix en haut à droite de la fenêtre : l'application se ferme.

B. Solution webapp (interface flask/heroku)

- Dataset

```
"intents": [
            {
                  "intent": "Greeting",
                  "text": [
                        "Hi",
                        "Hi there",
                        "Hola",
                        "Hello",
                        "Hello there",
                        "Hya",
                        "Hya there"
                  ],
                  "responses": [
                        "Hi human, please tell me your GeniSys user",
                        "Hello human, please tell me your GeniSys user",
                        "Hola human, please tell me your GeniSys user"
                  ],
```

Extrait du dataset Intent.json utilisé pour l'élaboration du chatbot ***Maredjo***

Un deuxième jeu de données a été fourni par le client au format JSON. Le document est divisé en *intent*, soit le but qu'à l'utilisateur lorsqu'il saisit sa question. Pour chaque *intent*, il est définit des *utterances* (déterminés ici par *text*), c'est-à-dire différents énoncés qu'un utilisateur peut taper pour atteindre son but (*intent*). Ces *utterances* sont accompagnées d'un set de réponses que le chatbot va retourner à l'utilisateur en accord avec l'*intent* identifiée.

Le jeu de données contient 22 intents, ayant chacune six utterances en moyenne et environ trois réponses. Les seules exceptions sont faites pour les *intents* "Gossip" et "Joke" qui possèdent plus de 1 000 réponses.

- Préparation des données

Comme pour la solution applicative, la ponctuation est retirée et le texte est converti en minuscules. Chaque mot est identifié et un numéro lui est associé. Un padding est ajouté pour que les entrées dans le réseau de neurones soient toutes de même longueur.

- Modèle

```
#creating model
model_sw = Sequential()
model_sw.add(Embedding(vocabulary+1,10))
model_sw.add(InputLayer(input_shape=(input_shape,)))
model_sw.add(SimpleRNN(30))
model_sw.add(Dense(output_length, activation="softmax"))
```

Le modèle adopté pour le notre chatbot est celui que vous voyez au-dessus. Il est composé d'une première couche d'embedding, suivi par l'input layer pour préciser la shape des inputs qui seront nécessaires pour obtenir les prédictions. Puis, une couche de SimpleRNN avec 30 unités, et enfin une couche Dense pour terminer l'architecture de notre modèle.

- Déroulé d'une conversation

Première étape, l'utilisateur arrive sur cette interface en rejoignant l'adresse de déploiement du chatbot réalisé par notre équipe : https://maredjo-v2-chatbot.herokuapp.com/

Une fois l'utilisateur en ligne, notre chatbot va lui envoyer un premier message pour signaler sa disponibilité et engager la conversation. Le nom du chatbot (modifiable selon les envies du client) ainsi que l'heure d'envoi du message figurent dans la bulle de conversation.

Deuxième étape, l'utilisateur entre une première phrase pour entamer la discussion avec le chatbot. Il utilise la ligne d'entrée grisée, sur lequel on retrouve le placeholder "Enter your message".

Une fois le message écrit, il suffit d'utiliser le bouton "send" en vert ou bien d'appuyer sur la touche entrée du clavier pour que le message soit envoyé au chatbot. Le message de l'utilisateur est retranscrit sur la fenêtre de dialogue, dans la bulle bleue à droite. On retrouve également le nom de l'utilisateur et l'heure d'envoi du message.

La conversation s'enchaîne tant que l'utilisateur continue de répondre et d'envoyer des messages, et la liste de tous les messages est consultable en remontant dans la conversation avec la souris ou le clavier.

Une fois l'utilisation terminée, il suffit de fermer l'application pour quitter le chatbot et mettre fin à la session.

5. Livrables et liens

A. Notebook EDA

Ce notebook contient le travail effectué par notre équipe pour l'exploration des données fournies pour entraîner notre modèle. Vous pouvez retrouver les différentes informations concernant les mots les plus présents, les différentes colonnes du dataset, les informations manquantes ou les valeurs aberrantes.

B. Notebook rnn_lstm_basedataset

Le notebook rnn_lstm_basedataset contient les différentes étapes concernant la préparation des données, ainsi que la mise en place de l'architecture du réseau de neurones de notre chatbot. Vous pouvez également trouver les modalités d'évaluation de nos différents modèles.

C. Fichier h5 pour l'export du modèle

Ce fichier contient le modèle final que nous avons utilisé pour faire fonctionner le chatbot demandé par le client. Il a été exporté au format h5 et contient l'architecture de notre modèle entraîné.

D. Lien des livrables et de l'application

Vous trouverez ci-dessous les liens des livrables ainsi que celui de l'application déployée sur heroku. Dans l'ordre : le dépôt github avec le code source, le modèle tkinter et enfin, l'application flask déployée.

- https://github.com/marianneSimplon/chatbot
- https://drive.google.com/file/d/1fvP0YZ4uZ7-PcfYaBk0SS_12kfJY0FZO/view?usp=sharing
- https://maredjo-v2-chatbot.herokuapp.com/

6. Axes d'amélioration

A. Amélioration du modèle

Le modèle utilisé dans la version déployée actuellement est un LSTM. Les résultats obtenus peuvent être améliorés avec la mise en place de couches différentes. Nous avons encore plusieurs pistes à explorer, avec notamment l'ajout de couches différentes ou de couches supplémentaires. Voici une liste non exhaustive des pistes prioritaires que nous voulons explorer :

- Ajout d'une couche LSTM en plus dans notre architecture existante
- Modification de la couche LSTM pour passer sur une couche Bidirectional(LSTM)
- Recherche des hyperparamètres les plus adaptés avec le fine tuning de notre modèle via Grid Search.

B. Amélioration de l'interface

Les deux solutions proposées par notre équipe disposent d'une interface utilisateur fonctionnelle, mais perfectible en ce qui concerne l'expérience utilisateur.

En ce qui concerne l'interface de la première solution (tkinter), une refonte graphique semble nécessaire afin de rendre l'utilisation de notre logiciel beaucoup plus agréable et intuitive. Notre équipe est en contact avec le pôle UI/UX de notre entreprise afin de voir ensemble les éléments essentiels à revoir avant la livraison de la prochaine version.

Il y a également un souci au niveau de l'accessibilité de notre logiciel. En effet, l'absence de lien entre l'application et le clavier de l'utilisateur rend son utilisation impossible pour beaucoup d'utilisateurs. Il faut donc ajouter l'utilisation de la touche "Enter" pour pouvoir envoyer le message, et non plus se baser uniquement sur le bouton "send" de l'interface. Ceci permettra aux utilisateurs sans souris d'utiliser notre chatbot.

Pour l'interface de notre application web, les emplacements pour l'avatar du chatbot et de l'utilisateur sont implémentés. Cependant, il n'est pas possible pour l'utilisateur d'ajouter son propre avatar ou bien de voir celui du chatbot. Le problème

vient du stockage des images sur notre application, et cette possibilité n'est pas encore prise en charge par la solution actuellement proposée.

Pour finir, il faudrait également avoir des retours d'utilisateurs sur les deux interfaces, afin de répondre aux attentes des futurs utilisateurs que nous n'avions pas prévu. Le passage par une phase de test sur un panel d'utilisateurs différents semble être la solution la plus adaptée pour réaliser cette amélioration.