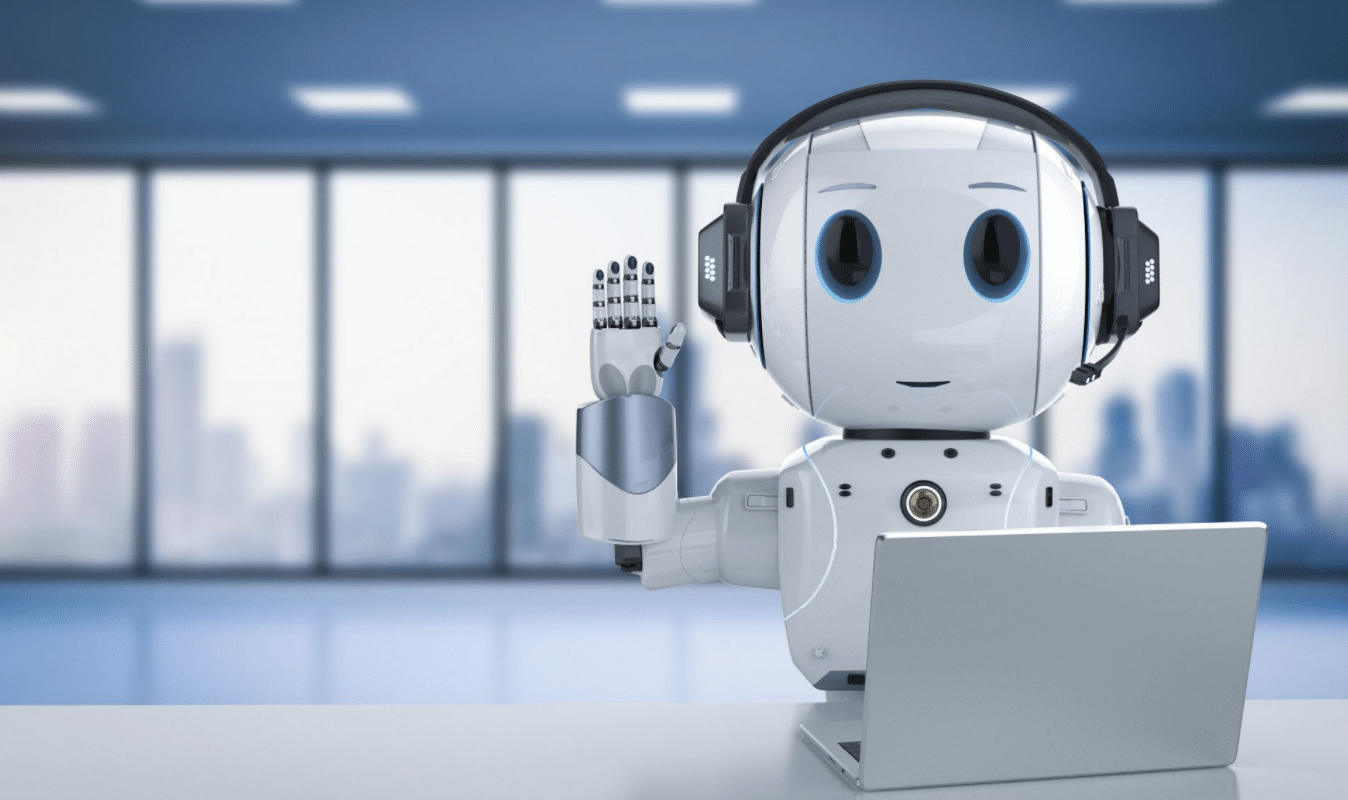
2022

RAPPORT

Projet CHATBOT



**Réalisation du projet ChatBot en utilisant le langage Python.**

**Travail réalisé par :**

*HINCHI Kawtar – KHANNOUCHI Oumaima – EL JAMAAI Mohamed – GARAAOUCH Safouan*

Sommaire

[**INTRODUCTION** 4](file:///C:\Users\hp\Downloads\RapportChatbot.docx#_Toc105540331)

[**Résumé** 5](#_Toc105540332)

[**Etude théorique** 6](#_Toc105540333)

[Présentation du Chatbot 6](#_Toc105540334)

[Choix du langage de programmation/logiciel 6](#_Toc105540335)

[Analyse et conception 6](#_Toc105540336)

[Présentation du Chatbot 7](#_Toc105540337)

[Choix du langage de programmation 8](#_Toc105540338)

[Choix du logiciel 9](#_Toc105540339)

[Analyse du Chatbot 10](#_Toc105540340)

[Conception 11](#_Toc105540341)

[**Etude pratique** 12](#_Toc105540342)

[Implémentation : 12](#_Toc105540343)

[Programme Python : 12](#_Toc105540344)

[Application de Travel Bot : 12](#_Toc105540345)

[**CONCLUSION** 14](file:///C:\Users\hp\Downloads\RapportChatbot.docx#_Toc105540346)

INTRODUCTION

L'intelligence artificielle (IA) est un processus d'imitation de l'intelligence humaine qui repose sur la création et l'application d'algorithmes exécutés dans un environnement informatique dynamique. Son but est de permettre à des ordinateurs de penser et d'agir comme des êtres humains.

Pour se rapprocher le plus possible du comportement humain, l'intelligence artificielle a besoin d'une quantité de données et d'une capacité de traitement élevées.

Ceci dit, c’est une discipline qui peut être qualifiée d’autonome et d’adaptative.

L’IA a dominé dans divers domaines : Résolution des problèmes, reconnaissance des voix, de l’écriture et de visage, le robotique, le traitement du langage naturel, et les systèmes experts. Dans ce rapport, nous allons principalement nous intéresser au système expert ; C’est un outil informatique d’intelligence artificielle, conçu pour simuler le savoir-faire d’un spécialiste humain, dans un domaine précis et bien délimité, grâce à l’exploitation d’un certain nombre de connaissances fournies explicitement par des experts du domaine. Les chatbots, ou autrement appelés les agents conversationnels, sont l’application directe de cet outil informatique.

Ce sont des assistants virtuels capables de simuler une conversation en imitant le comportement humain, que ce soit à l’écrit ou à l’oral. Il existe 2 types d’interfaces : les chatbot textes comme par exemple Messenger, et les interfaces voix qui permettent d’interagir de façon naturelle avec une machine. Le chatbot est un outil qui, grâce à la technologie moderne, est devenu nécessaire et efficace dans certains secteurs dont le tourisme ; En effet, les entreprises de voyages peuvent désormais communiquer avec des clients grâce à des robots utilisant des applications de chat, et développent ainsi mieux leur relation client.



**Résumé**

L’intelligence artificielle étant un sujet très vaste, notre projet se dirigera particulièrement à la création d’un chatbot qui sera capable de répondre à des questions/commandes simples, ou encore de planifier des rendez-vous.

***« Les efforts réalisés pour faire en sorte que l’Intelligence Artificielle soit sûre avaient 5 à 10% de chances de réussir. »***

*Elon Musk.*

Ce rapport contient donc principalement tout d’abord une explication détaillée du chatbot, suivie par la présentation du langage de programmation et du logiciel par lequel nous allons effectuer l’exécution du programme, nous

allons ensuite passer à l’étude pratique du projet où on expliquera les étapes suivies lors du codage du chatbot en Python.

Nous allons finalement présenter, sous forme de captures d’écran démonstratives, l’exécution du programme.

**Etude théorique**

Il est évident de réaliser une étude théorique avant de passer à l’étude pratique dans tout projet, afin d’avoir une idée plus claire du chatbot, de ses objectifs, sa conception, et des outils utilisés pour sa création ; Nous allons donc faire une brève présentation du Chatbot, suivie par l’explication du choix du langage de programmation (Python) et du logiciel de l’exécution du programme, et nous allons finir par une analyse du fonctionnement du chatbot, ainsi que sa conception.

*N° 01 -*

Présentation du Chatbot

Il s’agit de contextualiser l’idée du Chatbot que nous proposons dans le domaine du Tourisme, ses objectifs, son efficacité, ses exigences, et son rôle dans une agence de voyage.

*N° 02 -*

Choix du langage de programmation/logiciel

Nous avons choisi de faire un programme codé en python, un langage facile et considéré comme étant l’un des meilleurs langages de programmation pour les projets d’apprentissage automatique d’intelligence artificielle.

***N° 03 -***

Analyse et conception

En se basant sur les objectifs principaux et du but du chatbot relevant du concept de NLP (*Natural Language Processing*), nous allons exploiter les manières de son fonctionnement, ses étapes et sa finalité.

Présentation du Chatbot

Souvent, quand un visiteur trouve des difficultés à se faire une idée claire des services offerts par une agence de voyage via son site, il passe immédiatement à l’agence suivante, ce manque de clarté entraîne donc une énorme perte de trafic. Ceci dit, cette perte de trafic peut être évitée en créant des chatbots d'agences de voyages qui peuvent aider les clients à obtenir des informations détaillées et clairement présentées sur : Les chambres d'hôtel, les vols, les attractions, les locations de voitures, les forfaits pour les groupes, etc. Toute information supplémentaire recherchée par un utilisateur peut également être récupérée par le chatbot.

Un chatbot est un terme anglais qui désigne un robot conversationnel capable de donner l’illusion de tenir une discussion et de répondre à des questions de la part des internautes. Le chatbot a pour objectif de communiquer facilement des informations aux clients, il jouera en quelques sortes le rôle d’un agent de voyage en ligne qui tient compte des préférences de ses clients. Que ce soit à l’écrit ou à l’oral, il peut leur communiquer, par exemple, les horaires d’un avion. Plus encore, s’il a été correctement nourri en informations pertinentes, il leur prodiguera des conseils avisés pour choisir une destination en fonction des critères énoncés (budget, saison, envies, etc.).

Choix du langage de programmation

**Python :**

Le langage Python est un langage de programmation open source multi-plateformes et orienté objet. Grâce à des bibliothèques spécialisées, Python s'utilise pour de nombreuses situations comme le développement logiciel, l'analyse de données, ou la gestion d'infrastructures.

**Justification du choix :**

Python est un langage de programmation relativement simple à comprendre et donc à apprendre. Il suit une structure organisée avec une syntaxe simple et permet une utilisation des plus polyvalentes allant de simples applications web à des systèmes d’exploitation complets. Il est aujourd’hui considéré comme le langage le plus apprécié par les débutants en programmation.

Python est particulièrement apprécié par les développeurs d’application en sciences de données et en intelligence artificielle. En effet, Python est un langage qui prend en charge la gestion et la manipulation de données volumineuses et certaines de ses fonctions gèrent le traitement d’informations multiples de manière automatisée ainsi que la collecte et le nettoyage de données. Il permet notamment la modélisation de ces données et est très utilisé en Data Visualization.

De plus, grâce aux différents frameworks de tests, tels que les tests unitaires (unitests) ou pytests, il est possible d’automatiser des tests internes à la programmation et ainsi éviter les erreurs de codage. Ce qui est particulièrement utile et rassurant quand on est débutant.

Choix du logiciel

**VS Code :**

**Visual studio code** ou **VS Code**est un éditeur de code développé par **Microsoft** en 2015. Contrairement à ce à quoi Microsoft a eu l’habitude de nous habituer durant des années, il est l’un de ces premiers produits open source et gratuit, et surtout disponible sur les systèmes d’exploitation Windows, Linux et Mac. **Vs code** est développé avec le framework.

**Justification du choix :**

Facile à installer, à comprendre, à utiliser et rapide, il dispose d’une interface graphique responsive et customisable via des thèmes déjà installés.

Quel que soit le langage :  **VS code** permet de développer soit via les fonctionnalités par défaut ou en ajoutant des extensions disponibles selon les besoins de chacun.

Avec la fonctionnalité « IntelliSense », **VS Code** assiste les développeurs tout au long de leurs développements afin de simplifier l’édition du code, l’appel des propriétés, ou encore sur l’importation des dépendances. Ce système est disponible par défaut pour plusieurs langages mais aussi peut être configuré pour d’autres.

L’un des outils le plus important de **VS Code** est la capacité de déboguer les applications directement sur l’éditeur sans même avoir recours aux navigateurs, grâce à un système de points d’arrêt et une console de débogage intégrée qui permet de résoudre les problèmes directement dans l’éditeur.

Analyse du Chatbot

Le fonctionnement des chatbots est basé principalement sur un algorithme ayant comme finalité l’interaction avec l’utilisateur. A l’heure actuelle, il existe deux types de robots : Le premier agit dans le cadre défini par le créateur tandis que le second agit en fonction des échanges apportés par l’utilisateur.

Le but d’un chatbot est donc d’interagir avec le client en répondant à ses questions comme si c’était un ami “super intelligent”. Un chatbot peut réagir à la voix ou à des commandes écrites. Aussi bien à partir du texte que de la voix, le chatbot va décrypter les mots-clés de la requête pour y répondre. Plus le message sera précis et simple, plus il sera facile pour le chatbot d’analyser votre propos et donc de vous répondre. Vous obtiendrez des réponses utiles et personnalisées en fonction de vos questions.

D’abord, l’utilisateur utilise son clavier ou le micro pour poser sa question. On parlera ici d’interface. Deuxième étape, le chatbot reçoit l’information et l’analyse à travers une interface web ou l’interface d’une application. Cette étape de compréhension inclut la détection d’intention : “moteur” de notre chatbot. La troisième étape consiste à analyser la question. Il consulte sa base de données pour fournir la réponse la plus adaptée possible.

Pour résumer, vous avez d’un côté l’interface avec une reconnaissance d’intention ou de paramètres grâce au “machine learning“. Il est à noter qu’il faudra également gérer le “moteur” de votre chatbot en créant différents scénarios avec vos clients.

**A noter que :**

Tous les chatbots relèvent des concepts **NLP** (*Natural Language Processing*). La NLP comprend deux choses :

* NLU (Natural Language Understanding) : C’est la capacité des machines à comprendre le langage humain comme l’anglais. le français etc…
* NLG (Natural Language Generation): C’est la capacité d’une machine à générer du texte similaire à des phrases écrites par des humains.

Conception

**La structure du fichier Chatbot** :

Une fois le projet terminé on peut avoir une idée de la façon dont le projet sera mis en œuvre et cela en se basant sur la structure du fichier chatbot :

* **Train\_chatbot.py :** Dans ce fichier, nous allons créer et former le modèle de deep learning ou apprentissage profond. Ce dernier va classer et identifier ce que l’utilisateur demande au robot.
* **Chat.py :** C’est dans ce fichier que nous allons créer une interface utilisateur graphique pour tchatteravec notre chatbot formé.
* **Intents.json :** Le fichier d’intents contient toutes les données que nous allons utiliser pour former le modèle. Il comprend une collection de balises avec leurs modèles et réponses correspondants.
* **Chatbot\_model.h5 :** Il s’agit d’un fichier de format de données hiérarchique dans lequel nous avons sauvegardé les poids et l’architecture de notre modèle formé.
* **Classes.pkl :** Le fichier pickle peut être utilisé pour sauvegarder tous les noms de balises à classer lorsque nous prédisons le message.
* **Words.pkl :** Le fichier pickle words.pkl contient tous les mots uniques qui constituent le vocabulaire de notre modèle.

**Etude pratique**

Implémentation :

Dans ce projet Python, nous allons construire un chatbot en utilisant des techniques d'apprentissage en profondeur. Le chatbot sera formé sur l'ensemble de données qui contient **les catégories** (intentions), **le modèle** et **les réponses**. Nous utilisons un réseau neuronal récurrent spécial (LSTM) (*Long Short-Term Memory*) pour classer à quelle catégorie appartient le message de l'utilisateur, puis nous donnerons une réponse aléatoire à partir de la liste des réponses. Créons un chatbot basé sur la récupération en utilisant NLTK (*Natural Language ToolKit*), Keras, Python, etc. Le projet nécessite une bonne connaissance de Python, Keras et du traitement du langage naturel (NLTK). Simultanément, nous utiliserons des modules d'aide que vous pouvez télécharger à l'aide de la commande python-pip.

Programme Python :

Etape 1 :

Nous allons tout d’abord créer un nouveau fichier Python et le nommer train\_chatbot. Ensuite, nous allons importer tous les modules requis. Puis, nous allons lire le fichier de données JSON dans notre programme Python. Le fichier de données est au format JSON, nous avons donc utilisé le package json pour analyser le fichier JSON en Python.

Python

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from keras.optimizers import SGD

import random

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import json

import pickle

intents\_file = open('intents.json').read()

intents = json.loads(intents\_file)

Etape 2 :

Lorsque nous travaillons avec des données textuelles, nous devons effectuer divers prétraitements sur les données avant de créer un modèle d'apprentissage automatique ou d'apprentissage en profondeur. Sur la base des exigences, nous devons appliquer diverses opérations pour prétraiter les données. La tokenisation est la chose la plus basique et la première que vous puissiez faire sur les données textuelles. La tokenisation est le processus consistant à diviser l'ensemble du texte en petites parties comme des mots. Ici, nous parcourons les modèles et segmentons la phrase à l'aide de la fonction nltk.word\_tokenize() et ajoutons chaque mot dans la liste de mots. Nous créons également une liste de classes pour nos balises

for intent in intents['intents']:

 for pattern in intent['patterns']:

 # take each word and tokenize it

 w = nltk.word\_tokenize(pattern)

 words.extend(w)

 # adding documents

 documents.append((w, intent['tag']))

 # adding classes to our class list

 if intent['tag'] not in classes:

 classes.append(intent['tag'])

Etape 3 :

Maintenant, nous allons lemmatiser chaque mot et supprimer les mots en double de la liste. La lemmatisation est le processus de conversion d'un mot dans sa forme de lemme, puis de création d'un fichier pickle pour stocker les objets Python que nous utiliserons lors de la prédiction.

words = [lemmatizer.lemmatize(w.lower()) for w in

words if w not in ignore\_words]

words = sorted(list(set(words)))

classes = sorted(list(set(classes)))

print (len(documents), "documents")

print (len(classes), "classes", classes)

print (len(words), "unique lemmatized words",

words)

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(classes,open('classes.pkl','wb'))

Etape 4 :

A ce stade, nous allons créer les données d'entraînement dans lesquelles nous fournirons l'entrée et la sortie. Notre entrée sera le modèle et la sortie sera la classe à laquelle appartient notre modèle d'entrée. Mais l'ordinateur ne comprend pas le texte, nous allons donc convertir le texte en nombres

words = [lemmatizer.lemmatize(w.lower()) for w in

words if w not in ignore\_words]

words = sorted(list(set(words)))

classes = sorted(list(set(classes)))

print (len(documents), "documents")

print (len(classes), "classes", classes)

print (len(words), "unique lemmatized words",

words)

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(classes,open('classes.pkl','wb'))

Etape 5 :

Nous avons nos données d'entraînement prêtes, maintenant nous allons construire un réseau neuronal profond qui a 3 couches. Nous utilisons pour cela l'API séquentielle Keras. Après avoir entraîné le modèle pendant 200 époques, nous avons atteint une précision de 100 % sur notre modèle. Enregistrons le modèle sous le nom 'chatbot\_model.h5'

model = Sequential()

model.add(Dense(128, input\_shape=(len(train\_x[0]),),

activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train\_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov

accelerated gradient gives good results for this model

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,

nesterov=True)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model

hist = model.fit(np.array(train\_x), np.array(train\_y),

epochs=200, batch\_size=5, verbose=1)

model.save('chatbot\_model.h5', hist)

print("model created")

Etape 6 :

Pour prédire les phrases et obtenir une réponse de l'utilisateur pour nous laisser créer un nouveau fichier 'chatapp.py'. Nous chargerons le modèle entraîné, puis utiliserons une interface utilisateur graphique qui prédit la réponse du bot. Le modèle ne nous indiquera que la classe à laquelle il appartient, nous allons donc implémenter des fonctions qui identifieront la classe et nous récupéreront ensuite une réponse aléatoire dans la liste des réponses. Encore une fois, nous importons les packages nécessaires et chargeons les fichiers pickle 'words.pkl' et 'classes.pkl' que nous avons créés lors de l'entraînement de notre modèle :

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

from keras.models import load\_model

model = load\_model('chatbot\_model.h5')

import json

import random

intents = json.loads(open('intents.json').read())

words = pickle.load(open('words.pkl','rb'))

classes = pickle.load(open('classes.pkl','rb'))

Etape 7 :

Pour prédire la classe, nous devrons fournir des informations de la même manière que nous l'avons fait lors de la formation. Nous allons donc créer des fonctions qui effectueront un prétraitement du texte puis prédisent la classe.

def clean\_up\_sentence(sentence):

sentence\_words = nltk.word\_tokenize(sentence)

sentence\_words = [lemmatizer.lemmatize(word.lower()) for word in sentence\_words]

return sentence\_words

def bow(sentence, words, show\_details=True):

sentence\_words = clean\_up\_sentence(sentence)

# bag of words - matrix of N words, vocabulary matrix

bag = [0]\*len(words)

for s in sentence\_words:

for i,w in enumerate(words):

if w == s:

bag[i] = 1

if show\_details:

print ("found in bag: %s" % w)

return(np.array(bag))

def predict\_class(sentence, model):

p = bow(sentence, words,show\_details=False)

res = model.predict(np.array([p]))[0]

ERROR\_THRESHOLD = 0.25

results = [[i,r] for i,r in enumerate(res) if r>ERROR\_THRESHOLD]

results.sort(key=lambda x: x[1], reverse=True)

return\_list = []

for r in results:

return\_list.append({"intent": classes[r[0]], "probability": str(r[1])})

return return\_list

Etape 8 :

Nous allons maintenant développer une interface utilisateur graphique. Utilisons la bibliothèque Tkinter qui est livrée avec des tonnes de bibliothèques utiles pour l'interface graphique. Nous prendrons le message d'entrée de l'utilisateur, puis utiliserons les fonctions d'assistance que nous avons créées pour obtenir la réponse du bot et l'afficher sur l'interface graphique. Voici le code source complet de l'interface graphique

import tkinter

from tkinter import \*

def send():

 msg = EntryBox.get("1.0",'end-1c').strip()

 EntryBox.delete("0.0",END)

 if msg != '':

 ChatLog.config(state=NORMAL)

 ChatLog.insert(END, "You: " + msg + '\n\n')

 ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

 res = chatbot\_response(msg)

 ChatLog.insert(END, "Bot: " + res + '\n\n')

 ChatLog.config(state=DISABLED)

 ChatLog.yview(END)

base = Tk()

base.title("Flight Bot")

base.geometry("400x550")

base.resizable(width=FALSE, height=FALSE)

main\_menu = Menu(base, bg="RosyBrown3")

# Create the submenu

file\_menu = Menu(base, bg="RosyBrown3")

# Add commands to submenu

file\_menu.add\_command(label="New..")

file\_menu.add\_command(label="Save As..")

file\_menu.add\_command(label="Exit")

main\_menu.add\_cascade(label="File", menu=file\_menu)

#Add the rest of the menu options to the main menu

main\_menu.add\_command(label="Edit")

main\_menu.add\_command(label="Quit")

base.config(menu=main\_menu)

Application de Travel Bot :

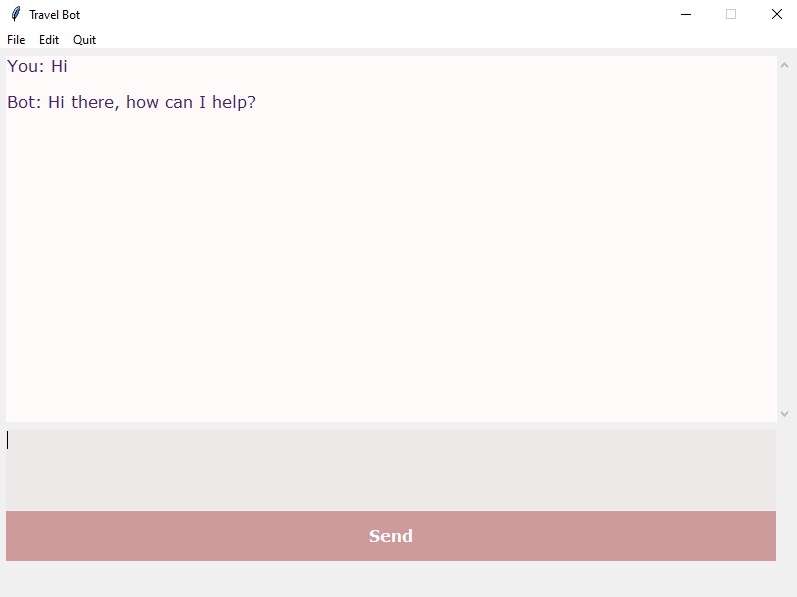
Pour exécuter le chatbot, nous avons deux fichiers principaux ; train\_chatbot.py et chat.py . Tout d'abord, nous formons le modèle à l'aide de la commande dans le terminal :

py train\_chatbot.py

Si nous ne voyons aucune erreur pendant l'entraînement, nous avons réussi à créer le modèle. Ensuite, pour exécuter l'application, nous exécutons le deuxième fichier :

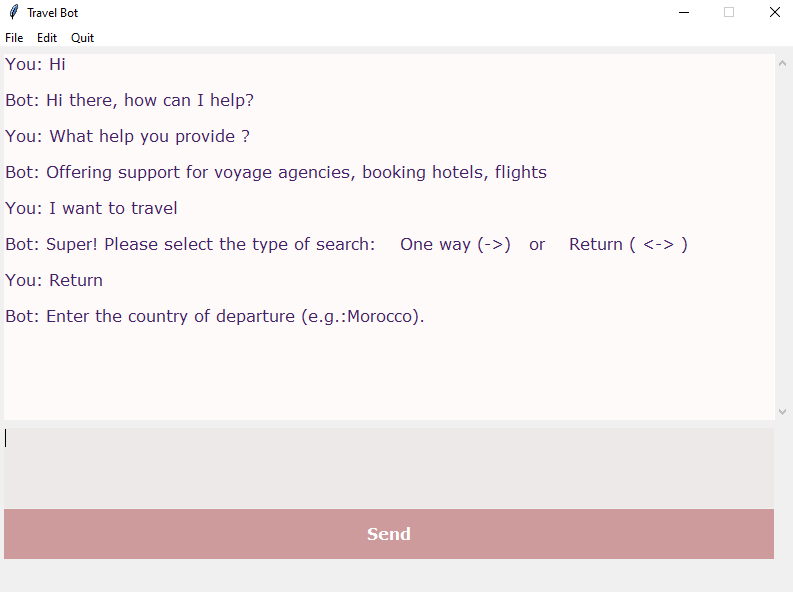
py chat.py

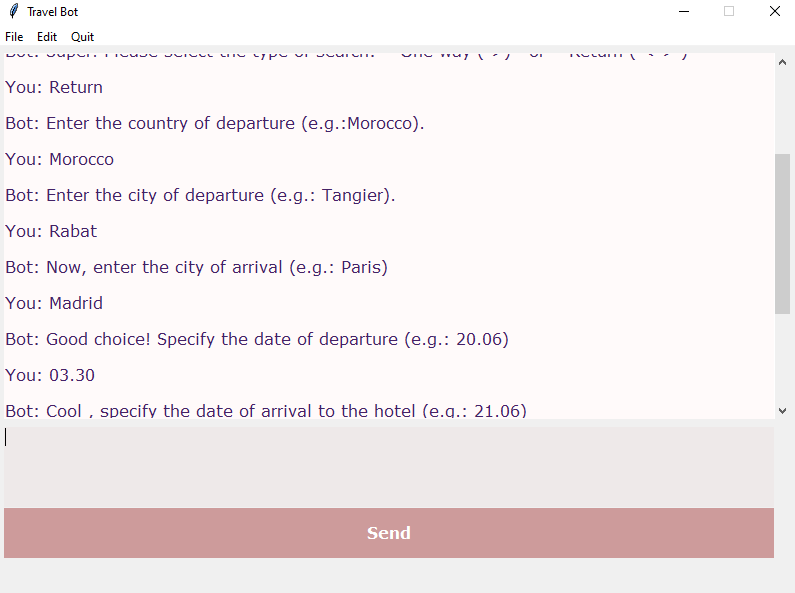
Le programme ouvrira une fenêtre GUI en quelques secondes. Avec l'interface graphique, on va pouvoir facilement discuter avec le bot.



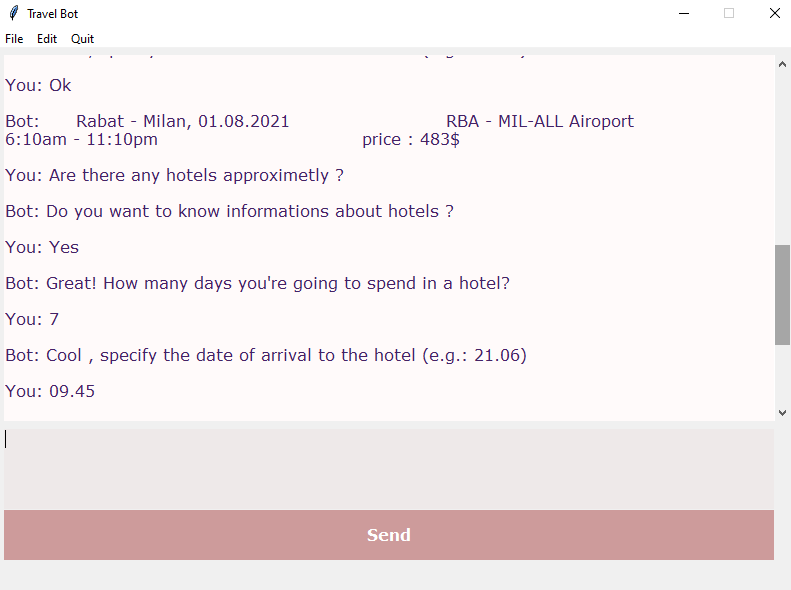
**Application du Travel bot :**

Afin de tester le fonctionnement de notre chatbot, on va supposer qu’une personne a envie de voyager de Rabat (Maroc) à Madrid (Spain ).Pour cela, elle a besoin des informations concernant le prix des vols, la disponibilité des hôtels et d’autres informations liées au voyage prévu.









**CONCLUSION**

Au terme de ce projet, nous tenons à souligner que le travail sur les chatbots nous a permis de perfectionner nos compétences dans ce domaine et sera un atout de taille pour notre vie professionnelle future, étant donné l’intérêt porté par les entreprises aux Chatbots de nos jours. Nous espérons pouvoir améliorer ce projet prochainement et qu’il trouvera de meilleures conditions pour entrer en vigueur. Ce fut une expérience riche et qui vaudra la peine d’être intégrée à nos CV.

***Nous vous remercions pour votre soutien continu***