

CHATBOT INTELLIGENT

Au profit de l'industrie touristique marocaine.

Membres du groupe :

- Yahya BENABDELMALEK.
- Ilyass ELGHAIB.
- Anas ELKACEMI.
- Aicha TLEMSANI .



Table des matières

A-	Introduction	2
B-	Présentation théorique du Chatbot :	2
1-	Thème	2
2-	Etude du besoin :	3
3-	Solution proposée et fonctionnalités à implémenter :	4
C-	Présentation technique du chatbot :	4
1-	Architecture du chatbot :	4
2-	Base de connaissance du chatbot :	5
	TAG	5
	QUESTIONS	5
	REPONSES	5
D-	Implémentation	6
1-	Outils de développement	6
2-	Explication du code	7
3-	Exécution du chatbot	16
	Conclusion	17

A- Introduction

L'innovation dans le tourisme est au cœur des préoccupations gouvernementales. En effet, d'un point de vue économique, l'industrie touristique au Maroc est un secteur économique important, il représente de 8% à 10 % du PIB du pays. Sur le plan touristique, le Maroc dispose d'atouts remarquables : son climat, sa culture spécifique, ses villes impériales et sa position géographique ainsi que son importante infrastructure touristique.

Mais malgré ces chiffres, la concurrence dans le secteur du tourisme est vive, avec l'arrivée des destinations émergentes. Des destinations qui disposent d'infrastructures modernes, mais surtout intelligentes et durables.

Ce projet vient en parallèle avec les orientations ci-dessus et a comme objectif d'aider à la modernisation de l'offre et de l'infrastructure hôtelière à travers la réalisation d'un agent conversationnel intelligent qui va faciliter l'interaction des clients, et va leur permettre d'avoir des informations depuis n'importe où et surtout n'importe quand.

Cet agent conversationnel est développé avec le langage Python à l'aide des bibliothèques d'intelligence artificielle et d'apprentissage automatique.

B- Présentation théorique du Chatbot :

1- Thème

Fini le temps où les voyageurs devaient se rendre auprès de leurs agences de voyages locales pour réserver des vols ou rechercher un hôtel approprié. Pour répondre à la demande toujours croissante du marché et attirer un client adepte du numérique, les hôtels continuent d'améliorer leurs services et cherchent des moyens de rendre les voyages encore plus pratiques. A travers ce rapport, nous avons détaillé les étapes de réalisation d'un agent conversationnel intelligent qui a pour but d'améliorer l'expérience des clients grâce à l'apprentissage automatique. En effet, Les chatbots jouent de plus en plus un rôle dans l'ensemble des opérations hôtelières, qu'il s'agisse de générer des réservations directes et d'augmenter la conversion de sites Web en aidant les hôteliers comme les concierges et les agents de réception à automatiser les demandes et les demandes de renseignements répétitives des clients.

Un chatbot est un programme conçu pour simuler une conversation humaine en utilisant l'intelligence artificielle. Après être devenus l'un des mots les plus à la mode de l'année dernière, les chatbots devraient perturber l'industrie du voyage et établir une nouvelle norme dans le domaine de la réservation mobile, les utilisateurs sont attirés par les chatbots, un outil d'achat disponible et personnalisé. En outre, un bon chatbot peut aider les utilisateurs à réduire les longues heures de recherche indécise.

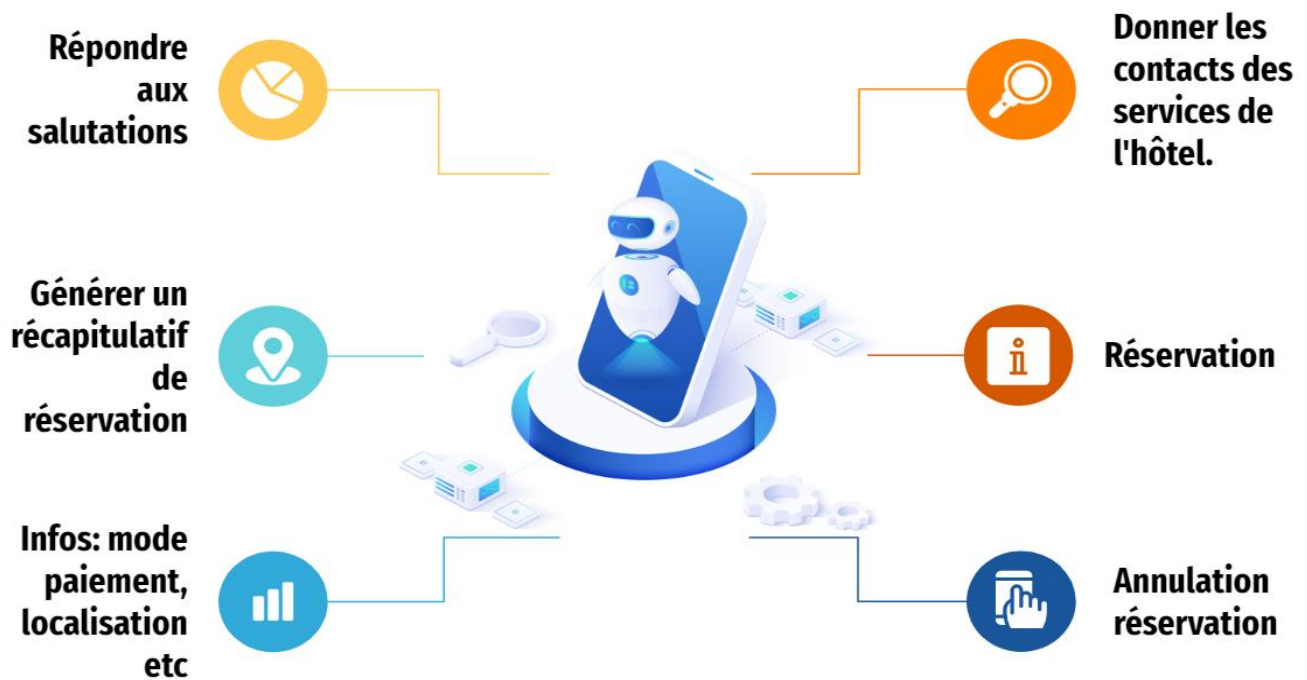
2- Etude du besoin :

Notre chatbot être un canal de d'échange d'information avec le client qui répondra aux besoins suivants :



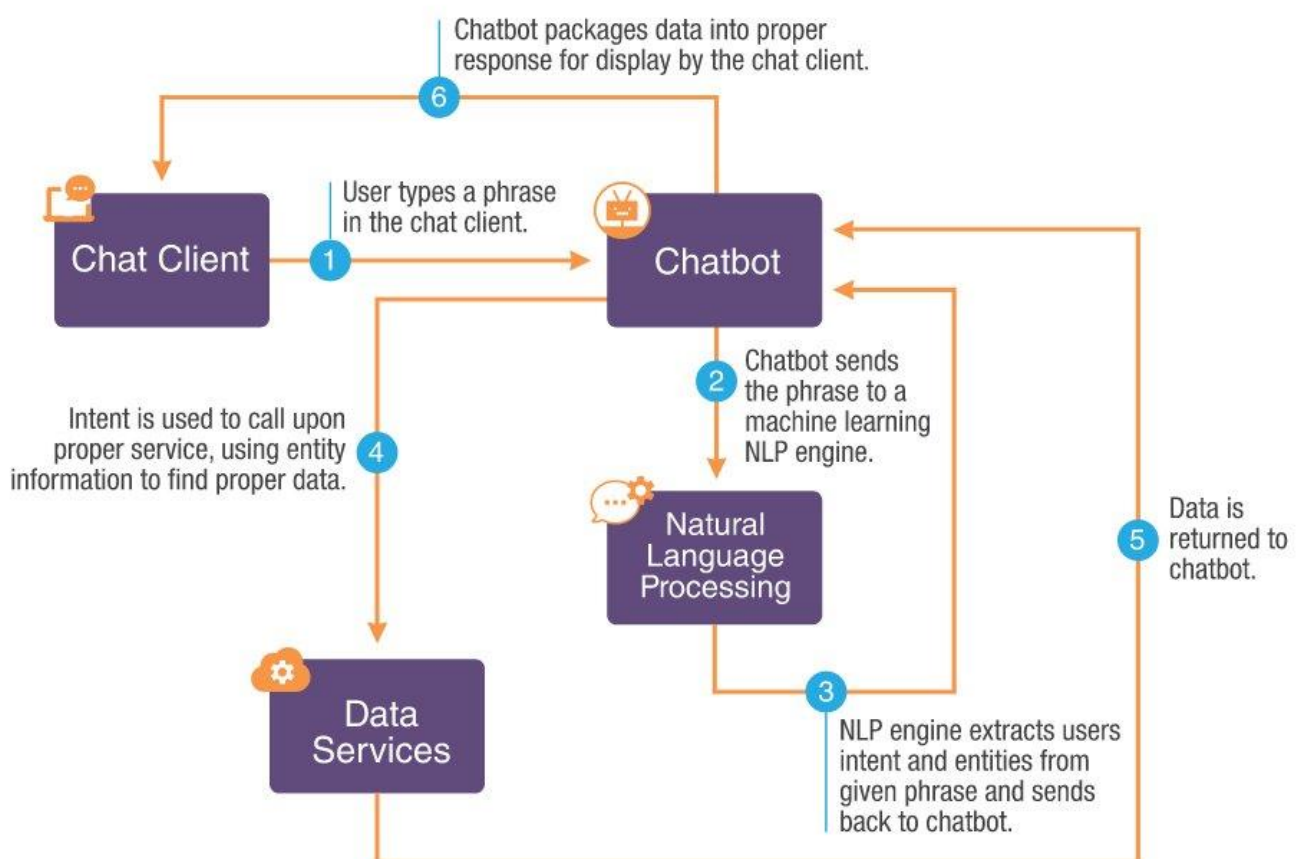
- **Gestion de réservation** : En termes d'hébergement, le chatbot va offrir au client des informations de réservation d'hospitalité
 - **Un besoin informationnel 24/7** : À partir de ce type de service, les clients peuvent être rapidement pris en charge avec les questions et les demandes les plus fréquentes qui ont été préalablement planifiées. Ce service peut répondre à la fois aux besoins de ceux qui n'ont pas encore de réservation, ainsi qu'aux clients qui sont sur place et qui ont besoin d'informations.
 - **Capacité de répondre au nombre de demandes illimité** : L'augmentation de la demande d'aide peut entraîner le besoin de plus d'employés et, par conséquent, des coûts plus élevés. Par contre, l'utilisation du chatbot nous permet de servir toutes ces personnes avec beaucoup moins d'efforts et de manière illimitée, car peu importe le nombre de personnes qui parlent au robot en même temps.
 - **L'immédiateté** : Le client enverra un message et recevra une réponse instantanée. En fait, avec ces réponses en temps réel, il y a de grandes chances que le client soit encore plus intéressé et admiré par l'efficacité du service de l'entreprise.
- « 64 % des voyageurs et 80 % des utilisateurs professionnels s'attendent à ce que les agences de voyages leur répondent en temps réel. » (Salesforce)
- **L'adaptabilité** : les utilisateurs peuvent poser des questions dans leurs propres mots et obtenir une réponse plutôt que de passer du temps à chercher ces réponses sur votre site Web.

3- Solution proposée et fonctionnalités à implémenter :



C- Présentation technique du chatbot :

1- Architecture du chatbot :



Nous allons construire un chatbot en utilisant des techniques d'apprentissage. Le chatbot sera formé sur l'ensemble de données qui contiennent les catégories (intentions), le modèle et les réponses. Nous utilisons un réseau neuronal récurrent spécial (LSTM) pour classer à quelle catégorie appartient le message de l'utilisateur, puis nous donnerons une réponse aléatoire à partir de la liste des réponses.

La création du chatbot sera basée sur la récupération en utilisant les modules NLTK, Python, etc.

2- Base de connaissance du chatbot :

TAG	QUESTIONS	REPONSES
Salutations	Hello Is anyone there Good day	Hello, thanks for visiting Hi there, what can I do for you Hi there, how can I help
Au revoir	Bye See you later Goodbye	See you later, thanks for visiting Have a nice day Bye! Come back again soon
Remerciement	Thank you That's helpful Thanks a lot!	Happy to help! Anytime! My pleasure
Annulation	I would like to cancel my booking I want to make a cancellation Can I cancel a room?	Sure, which is your reference number? I only need to know the reference number of your booking
Paie ment	When do I have to pay? Can I pay by card? Do you accept cash?	The payment is on arrival. We accept cash and card You can pay on arrival by cash or card
Location	Where is the hotel located? Can you give me the address? I want to know the exact location	The Marrakech Hotel is located at Loudaya St, Rabat , Morocco 2441 You will find us at Loudaya St, Rabat , Morocco 2441
Contacts	I would like to contact the marketing department I want to speak with someone from sales Can I have the email of finance?	Sure. To contact with them, send an email to:accountancy@marrakechhotel.com Of course. The email to contact with the department is:accountancy@marrakechhotel.com
Reservation	I would like to make a reservation Can I book a room? I want to make a booking	Sure, I only need to know a few details Lovely. Let's begin with the reservation

D- Implémentation

1- Outils de développement

Pour développer ce chatbot, on s'est servi du langage python en raison de la diversité des bibliothèques qu'il offre et qui traitent de l'intelligence artificielle et du Machine Learning. De plus, en utilisant Python, il nous offre la possibilité de développer un chatbot ayant une application GUI, c'est-à-dire créé et porté sur de nombreux systèmes d'appels de bibliothèques et systèmes Windows.

On s'est appuyé sur une variété de bibliothèques et des packages pour la réalisation de ce chatbot, principalement des bibliothèques du NLP (Natural Language Processing).

Nltk	Bibliothèque Natural Language Toolkit pour le traitement automatique des langues
Nltk.download('punkt')	Ce tokenizer ¹ divise un texte en une liste de phrases en utilisant un algorithme non supervisé pour créer un modèle pour les mots d'abréviation, les collocations et les mots qui commencent des phrases.
Nltk.stem.porter	C'est algorithme pour le décapage de suffixe, il nous permet de prendre que la racine du mot
TensorFlow	Est une bibliothèque de Machine Learning, il s'agit d'une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance.
Torch	Permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond (deep Learning).
Torch.utils.data	Il représente un Python itérable sur un jeu de données.
Torch.nn	Modules qui permettent de créer et de former des réseaux de neurones.
Pandas	Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données.

¹ Tokenizer : Le Tokenizer est un analyseur lexical, il permet, de tokenizer du code, c'est à dire transformer du code en liste tokens. Cette notion va être bien détaillé par la suite

NumPy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

2- Explication du code

```
[ ] # Import packages
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
import tensorflow as tf
import nltk
nltk.download('punkt')
from nltk.stem.porter import PorterStemmer
from datetime import datetime
import random
import string
import pandas as pd
```

En premier lieu, on a préparé l'environnement de développement par l'importation des bibliothèques qu'on va utiliser par la suite.

```
[ ] # Create a dictionary with your training data
intents = {
    'intents': [
        {
            'tag': 'greeting',
            'patterns': [
                'Hi',
                'Hey',
                'Hello',
                'How are you',
                'Is anyone there?',
                'Good day'
            ],
            'responses': [
                'Hey',
                'Hello, thanks for visiting',
                'Hi there, what can I do for you?',
                'Hi there, how can I help?'
            ]
        }
    ]
}
```

Dans cette partie on crée une base de connaissances qui va nous servir pour entraîner notre modèle machine learning du chatbot, pour cela, on a créé un dictionnaire appelé intents, de clé intents qui a pour valeur une liste des dictionnaires, ces dictionnaires sont composés de trois clés {tag, patterns, responses}.


```
[ ] stemmer = PorterStemmer()
def tokenize(sentence):
    """
    This function takes a sentence as an input,
    and returns a list of its tokens
    """
    return nltk.word_tokenize(sentence)

def bag_of_words(tokenized_sentence, all_words):
    """
    Function to represent a sentence into a vector of float numbers
    input: list of tokens in a sent and a list of all the words in the text
    output: vector equal to the vocab length for each sentence
    """
    tokenized_sentence = [stemmer.stem(w.lower()) for w in tokenized_sentence]
    bag = np.zeros(len(all_words), dtype=np.float32)

    for idx, w in enumerate(all_words):
        if w in tokenized_sentence:
            bag[idx] = 1.0

    return bag
```

Avant de créer notre modèle d'apprentissage automatique, on doit d'abord passer par une étape de prétraitement des données textuelles en jeu. Cette étape fait appel à plusieurs opérations et notions qu'on doit les expliquer tout d'abord.

- **Tokenization** : c'est un fractionnement d'une chaîne en unités significatives (par exemple, mots, caractères de ponctuation, chiffres). C'est-à-dire transformer du code en liste tokens. La tokenization, c'est la première étape de la compilation ou de l'interprétation de la plupart des langages informatiques. Prenons Python par exemple, l'ordinateur ne sait absolument pas quoi faire avec le fichier qu'on lui donne, il le découpe donc pour avoir chacun des mots du code et pouvoir comprendre ce qu'on lui demande.
- **Stemming** : La radicalisation est le processus de réduction d'un mot à sa racine de mot qui se fixe aux suffixes et aux préfixes ou aux racines des mots connus sous le nom de lemme. Par exemple : les mots "Likes", "liked", "likely" et "liking" va être radicaliser en "like".

Nous ne pouvons pas simplement transmettre la phrase d'entrée telle quelle à notre réseau de neurones. Nous devons en quelque sorte convertir les chaînes de modèle en nombres que le réseau peut comprendre. Et pour ce faire, nous convertissons chaque phrase en un soi-disant sac de mots (bag_of_words).

- **Bag_of_words** : Pour ce faire, nous devons collecter des mots d'entraînement, c'est-à-dire. C'est-à-dire tous les mots que notre bot peut consulter dans les données d'entraînement. Sur la base de tous ces mots, nous pouvons ensuite calculer le sac de mots pour chaque nouvelle phrase. Un sac de mots à la même taille que le tableau de tous les mots, et chaque position contient un 1 si le mot est disponible dans la phrase entrante, ou 0 sinon.

Training Data

bag of words

all words

["Hi", "How", "are", "you", "bye", "see", "later"]

"Hi"	→	[1, 0, 0, 0, 0, 0, 0]	0 (greeting)
"How are you?"	→	[0, 1, 1, 1, 0, 0, 0]	
"Bye"	→	[0, 0, 0, 0, 1, 0, 0]	1 (goodbye)
"See you later"	→	[0, 0, 0, 1, 0, 1, 1]	

X

y

Passant maintenant à expliquer quelque ligne de code qui peuvent apparaitre ambiguë.

- **PorterStemmer** : Cette fonction sert à initialiser le stemmer.
- **Word_tokenize** : Renvoie la liste des syllabes des mots, ou renvoie les mots d'une phrase.
- **Stemmer.stem** : c'est elle qui nous donne comme output le radical du mot

```
stemmer. stem("computer")
```

Output:

```
"comput"
```

- **w.lower**: La méthode lower () renvoie une chaîne où tous les caractères sont en minuscules.
- **np.zeros(len(all_words), dtype=np.float32)** : Matrice remplie de zéros de la taille de tous les mots et de type réelle
- **Enumerate** : La méthode enumerate() ajoute un compteur à l'itérable. L'objet renvoyé est un objet d'énumération

```
days= { 'Mon', 'Tue', 'Wed','Thu'}  
# le compteur ici commence de 5  
enum_days = enumerate(days, 5)  
print(list(enum_days))  
#result
```

```
[(5, 'Tue'), (6, 'Thu'), (7, 'Mon'), (8, 'Wed')]
```

```
[ ] all_words = []
    tags = []
    xy = []

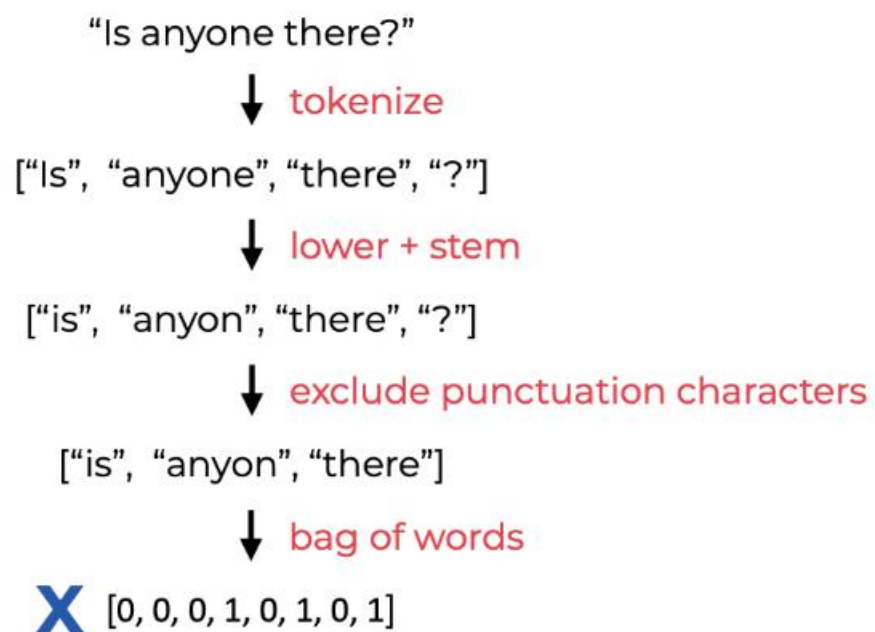
    # Save all the keywords in different variables
    for intent in intents['intents']:
        tag= intent['tag']
        tags.append(tag)
        for pattern in intent['patterns']:
            w = tokenize(pattern)
            all_words.extend(w)
            xy.append((w, tag))

    # Create the vocabulary
    ignore_words = ['?', '!', '.', ',']
    all_words = [stemmer.stem(w.lower()) for w in all_words if w not in ignore_words]
    all_words = sorted(set(all_words))
    tags = sorted(set(tags))
```

Cette partie du code prends les tags et les stocke dans une liste de tag, les racines des mots et les stocke dans all_words, et un couple (mot, tag correspondant) dans la liste xy.

Après on essaie de se débarrasser des points de ponctuation pour ne garder que les mots ayant un sens, et supprimer les doublants grâce à la fonction set qui nous donne comme résultat un dictionnaire qui englobe la totalité des mots. Ensuite, nous trions les mots et les tags par ordre alphabétique, puis utilisons l'index comme étiquette de classe. L'ensemble de notre pipeline de prétraitement ressemble à ceci :

Our NLP Preprocessing Pipeline



```
[ ] # Set the final training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

Préparer et déposer nos données d'entraînement après avoir passé par l'étape du pré-traitement.

```
class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    def __len__(self):
        return self.n_samples
```

```
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax
        return out
```

Cette étape désigne la mise en œuvre le réseau neuronal avec deux couches cachées.

Premièrement pour construire un réseau neuronal on doit générer une sortie à partir des données d'entrée. On se permet de le faire grâce à `nn.Linear`.

- **`super(NeuralNet, self).__init__()`** : Fait référence à la classe mère `nn.Module` car `NeuralNet` hérite de cette classe.
- **`Nn.Linear`** : `nn.linear` est un module utilisé pour créer un réseau d'avance à couche unique avec `n` entrées et `m` sorties.

Dans notre cas:

- `nn.Linear(input_size, hidden_size)` crée une couche de la taille `hidden_size` à partir d'une taille d'entrée égale à `input_size` et ainsi de suite pour les autres deux couches .

```

▶ # Hyperparameters
batch_size=8
hidden_size=8
output_size=len(tags)
input_size = len(X_train[0])
learning_rate = 0.001
num_epochs = 1000

dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=True, num_workers=2)

model = NeuralNet(input_size, hidden_size, output_size).to(device)

[ ] # Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

```

L'apprentissage profond ou Deep Learning traite souvent de grandes quantités de données. Ces données sont décomposées en plus petits morceaux (appelés lots ou **Batch**) et transmises aux réseaux de neurones un par un.

Dans le Deep Learning, on fait appelle aussi à la notion d'époch ou époque. Et une époque est celle où l'ensemble de données est passé en avant et en arrière à travers le réseau neuronal une fois. Afin de généraliser le modèle, nous utilisons plus d'une époque dans la majorité des modèles d'apprentissage profond.

Plus le nombre d'époques est élevé, plus les paramètres sont ajustés, ce qui donne un modèle plus performant. Cependant, trop d'époques pourraient conduire à un surajustement. Si un modèle est suréquipé, il fonctionne bien dans les données du train et fonctionne mal sur les données d'essai.

Pour cela , on aura besoin des hyperparamètres comme:

- **Batch_size** : le nombre d'échantillons qui seront transmis au réseau en même temps.
- **Num_epoch** : le nombre d'époch ou d'époque
- **Hidden_size** : la taille de la couche intermédiaire ou cachée
- **Input_size** :
- **Output_size** :

```
[ ] for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(device)

        # Forward
        outputs = model(words)
        loss = criterion(outputs, labels)

        # Backward and optimizer
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch + 1) % 100 == 0:
        print(f'epoch {epoch + 1}/{num_epochs}, loss = {loss.item():.4f}')

    print(f'final loss, loss = {loss.item():.4f}')
```

Entrainement du modèle pendant 1000 époque pour atteindre une meilleure précision de notre modèle.

```
[ ] all_bookings = pd.DataFrame()

class Customer:

    rate = 99.00

    def __init__(self, name, dates, room, service):
        self.name = name
        self.dates = (datetime.strptime(dates[0], '%d/%m/%Y').date(), datetime.strptime(dates[1], '%d/%m/%Y').date())
        self.room = room
        self.service = service

    def ID(self):
        letters = string.ascii_uppercase
        digits = string.digits
        a = random.sample(letters, 3) + random.sample(digits, 4)
        self.id = ''.join(a)
```

```

def nights(self):
    nights = (self.dates[1] - self.dates[0]).days
    return nights

def final_price(self):
    price = self.rate * float(self.nights())
    return price

def __str__(self):
    return f'''
    > Mr./Miss. {self.name[1]},
    >
    > We are delighted to confirm your booking with us for the {self.dates[0]} till the {self.dates[1]}
    > A {self.room} with {self.service} for the final rate of {self.rate}DH per night.
    > Total price: {self.final_price()}DH
    > Your reference number is {self.id}.
    > Keep this number in case you want to modify or cancel your booking in the future.
    >
    > Best,
    > The Marrakech Hotel
    '''

```

La class customer contient des méthodes qui calculent le prix total de la réservation du client, la construction d'une ID et l'affichage de la confirmation de la réservation avec toutes les caractéristiques que le client a choisie.

```

[ ] departments = {
    'marketing': ['marketing', 'seo', 'community manager'],
    'sales': ['reservations', 'sales', 'booking'],
    'accountancy': ['accountancy', 'finance', 'purchase']
}

```

```

[ ] def contact_dept(user_sent, departments):
    '''
    Takes the sentence and all the departments as input
    and returns the department email that the user wants to be contact with.
    '''
    email = None
    for k,v in departments.items():
        for d in user_sent:
            if d in v:
                email = f'{k}@marrakechhotel.com'
    return email

```

Ce bout de code contient la base de connaissance de différent département dans l'hôtel et une fonction qui permet d'afficher l'email d'un département demandé par le client.

```

imodel_dic = model.state_dict()
model = NeuralNet(input_size, hidden_size, output_size)
model.load_state_dict(model_dic)
model.eval()

bot_name = 'Bot '
print("Let's chat: type 'quit' to exit")

while True:
    sentence = input('You: ')
    if sentence == 'quit':
        break

    sentence = tokenize(sentence)
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
    X = torch.from_numpy(X)

    output = model(X)
    _, predicted = torch.max(output, dim=1)
    tag = tags[predicted.item()]

    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]

    if prob.item() > 0.75:
        for intent in intents['intents']:
            if tag == intent['tag']:
                if tag == 'booking':
                    print(f"{bot_name}: {random.choice(intent['responses'])}")
                    # Stage 1: Customer's Name
                    f_name = input('\tFirst Name: ')
                    l_name = input('\tLast Name: ')

                    # Stage 2: Booking Dates
                    arr = input('\tArrival day (DD/MM/YYYY): ')
                    dep = input('\tDeparture day (DD/MM/YYYY): ')

                    # Stage 3: Room and service
                    room = input('\tWhich type of room are you looking for?: ')
                    service = input('\tWhich service do you prefer?: ')

                    # Stage 4: Confirmation and Final Rate
                    c1 = Customer((f_name, l_name), (arr, dep), room, service)
                    c1.ID()
                    all_bookings.append(c1.__dict__, ignore_index=True)
                    print(c1)


                elif tag == 'cancellation':
                    print(f"{bot_name}: {random.choice(intent['responses'])}")
                    ref_num = input('\tReference number: ')
                    if ref_num in all_bookings['id'].values:
                        all_bookings = all_bookings.drop(all_bookings['id'][all_bookings['id'] == ref_num].index)
                        print('Your reservation has been canceled.')
                    else:
                        print('This reference number does not exist.')


                elif tag == 'contacts':
                    contact_email = contact_dept(sentence, departments)
                    if contact_email != None:
                        print(f"{bot_name}: {random.choice(intent['responses'])} {contact_email}")
                    else:
                        print('Unfortunately this department does not exist.')
                else:
                    print(f"{bot_name}: {random.choice(intent['responses'])}")
    else:
        print(f"{bot_name}: I do not understand...")

```

Ce code charge le modèle entraîné et fait des prédictions pour les nouvelles phrases.

3- Exécution du chatbot

 Let's chat: type 'quit' to exit



You: hey

Bot : Hi there, what can I do for you?

You: i want to make a reservation

Bot : Lovely. Let's begin with the reservation

First Name: Anas

Last Name: ELKACEMI

Arrival day (DD/MM/YYYY): 09/06/2022

Departure day (DD/MM/YYYY): 12/06/2022

Which type of room are you looking for?: Double

Which service do you prefer?: All included

> Mr./Miss. ELKACEMI,

>

> We are delighted to confirm your booking with us for the 2022-06-09 till the 2022-06-12.

> A Double with All included for the final rate of 99.0DH per night.

> Total price: 297.0DH

> Your reference number is DTZ9157.

> Keep this number in case you want to modify or cancel your booking in the future.

>

> Best,

> The Marrakech Hotel

You: thank you

Bot : Anytime!

You: what are type of payment

Bot : The payment is on arrival. We accept cash and card

You: i want to knwo the location of the hotel

Bot : You will find us at Loudaya St, Rabat , Morocco 2441

You: ca, i have the email of finance

You: can i have the email of finance

Bot : Sure. To contact with them, send an email to: accountancy@marrakechhotel.com

You: sorry i want to cancel my reservation

Bot : I only need to know the reference number of your booking

Reference number: DTZ9157

Your reservation has been canceled.

You:

Conclusion

Dans ce projet, nous avons compris les chatbots et implémenté une version d'apprentissage d'un chatbot en Python qui est précis. Les chatbots sont utilisés partout et toutes les entreprises sont impatientes d'implémenter des bots dans leur flux de travail. Nous, avons décidé de l'implémenter au niveau d'une structure hôtelière. La raison qui explique notre choix d'un chatbot intelligent implémenté au niveau d'un hôtel répond, en effet, à une problématique qui consiste à l'émergence de nouvelle destination touristiques intelligentes, ce qui impose le développement et la modernisation de la structure hôtelière marocaine .