UNIVERSITÉ MOHAMMED V

Faculté des Sciences Rabat

Master Ingénierie de Données et Développement Logiciel

Rapport fin de module NLP

Professeur : Abdelhak Mahmoudi

Réalisé par :

- Hala Bahida
- Fatima Bouya

Année universitaire 2020-2021

Introduction :Projet Question answering

Le traitement du langage naturel (NLP) est un domaine en pleine croissance au sein de l'apprentissage automatique et de l'intelligence artificielle. En termes simples, il s'agit d'apprendre aux machines à lire, comprendre et traiter les langues humaines. Un projet NLP peut avoir des centaines d'applications à travers la recherche, la vérification orthographique, la correction automatique, les chatbots, les recommandations de produits...

Le projet Question answering est un approfondissement du travail que nous avons réalisé lors du projet tutoré du module NLP du deuxième semestre, dirigé par Abdelhak Mahmoudi. Chatbot est un projet (patients conversationnelles guider et conseiller l'utilisateur et lui permettre de prendre des rendez-vous et de chercher des informations selon selon ses attentes) qui se divise en trois grandes phases de travail. Premièrement, une phase d'étude de technologies puis la conception et le développement de l'application

et finalement la conception et le développement de l'interface utilisateur. Notre choix de prolonger le projet entrepris au troisième semestre se justifie de l'intérêt que j'ai porté aux problématiques que posait le sujet, mais également du fait qu'aussi bien dans des domaines commerciaux, médicaux que de la recherche, l'intelligence artificielle est un phénomène d'actualité qui ne cesse d'évoluer. Les agents conversationnels incontournables pour de nombreux dispositifs et deviennent une alternative très souvent envisagée à certaines interfaces.

Chapitre 1: Préparation de l'environnement.

1. La définition d'un Chatbot:

Chatbot est un logiciel intelligent capable de communiquer et d'effectuer des actions similaires à celles d'un humain. Les chatbots sont beaucoup utilisés dans l'interaction client, le marketing sur les sites des réseaux sociaux et la messagerie instantanée du client. Il existe deux bases de models en fonction de la façon dont elles sont construites, modèles basé sur la récupération et basé sur la génération.

1-1. Chatbots basés sur la récupération:

Un chatbot basé sur la récupération utilise des modèles d'entrée et des réponses prédéfinis. Il utilise ensuite un certain type d'approche heuristique pour sélectionner la réponse appropriée. Il est largement utilisé dans l'industrie pour créer des chatbots axés sur les objectifs où nous pouvons personnaliser le ton et le flux du chatbot pour conduire nos clients avec la meilleure expérience.

1-2. Chatbots génératifs:

Les modèles génératifs ne sont pas basés sur des réponses prédéfinies.
Ils sont basés sur des réseaux de neurones seq 2 seq. C'est la même idée que la traduction automatique. En traduction automatique, on traduit le code source d'une langue à une autre langue mais ici, on va transformer l'entrée en sortie. Il a besoin d'une grande quantité de données et il est basé sur des réseaux Deep Neural.

2.Travail Effectué

2-1. Matériel utilisés:

L'ensemble du projet a été réalisé sur un ordinateur portable de marque hp elitebook ,core 17 (4eme génération) 16G RAM système d'exploitation windows 10 64 bits.

2-2. Technologie Utilisés :

Python: Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.

2-3. Logiciels utilisés:

Anaconda Navigator:Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda contient **Jupyter**, qui est essentiellement une combinaison entre un IDE et un serveur pour exécuter vos Notebooks. Jupyter prend en charge aujourd'hui plus de 40 langages informatiques.

Chapitre 2: La démarche de projet question answering

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 **recurrent neural network (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.

Le projet nécessite la connaissance de Python, Keras et du traitement du langage naturel (NLTK) .Avec eux, nous utiliserons des modules d'aide.

1. les bibliothèques et les modules utilisés:

Nltk :est une **bibliothèque** Python dédiée au traitement naturel du langage ou Natural Language Processing. Découvrez tout ce que vous devez savoir pour maîtriser cet outil.

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. Elle permet aux chercheurs de développer des architectures d'apprentissage expérimentales et de les transformer en logiciels.

keras: bibliothèque en open source, est un logiciel multiplateforme intégrant différentes fonctionnalités de développement et d'interaction pour les programmeurs. Conçu en langage Python, le système est exploité dans le domaine de l'intelligence artificielle (IA), particulièrement pour le deep learning.

Pickle: est un **module** de python qui permet de sauvegarder une ou plusieurs variables dans un fichier et de récupérer leurs valeurs ultérieurement. Les variables peuvent être de type quelconque.

pour le téléchargement de ces bibliothèques il suffit d'utiliser la commande:

- ➢ **pip install tensorflow, keras, pickle, nltk**

2. Les étapes à suivre durant la création de notre chatbot .

2-1. Préparation de la base de données.

L'ensemble de données que nous utiliserons est " **intents.json** ". Il s'agit d'un fichier JSON qui contient les modèles que nous devons rechercher et les réponses que nous souhaitons renvoyer à l'utilisateur.

```
{"intents": [
             {"tag": "greeting",
              "patterns": ["Bonjour","Bonne journée","Allô", "Bonsoir", "Salut", "ça roule?"],
              "responses": ["Salut à toi!", "Comment vas tu?", "Salutations!", "Enchanté"]
             },
             {"tag": "age",
              "patterns": ["Quel âge as-tu?", "C'est quand ton anniversaire?", "Quand es-tu né?"],
              "responses": ["J'ai deux mois", "Je suis né en 2021",
                           "Ma date d'anniversaire est le 3 juillet et je suis né en 1996", "03/07/2021"]
             },
             {"tag": "date",
              "patterns": ["Que fais-tu ce week-end?",
                "Tu veux qu'on fasse un truc ensemble?",
                 "Quels sont tes plans pour cette semaine"],
              "responses": ["Je suis libre toute la semaine", "Je n'ai rien de prévu", "Je ne suis pas occupé"]
             },
             {"tag": "name",
              "patterns": ["Quel est ton prénom?", "Comment tu t'appelles?", "Qui es-tu?"],
              "responses": ["Mon prénom est Chatty", "Je suis Chatty", "Chatty"]
             },
             {"tag": "goodbye",
              "patterns": [ "bye","Bonne soirée","À tout à l'heure ","Bonne nuit", "au revoir", "à bientot"],
              "responses": ["C'était sympa de te parler", "à plus tard", "On se reparle très vite!"]
             },
```

2-2. Importez et chargez le fichier de données.

Tout d'abord, créez un nom de fichier comme chatbot.py. Nous importons les packages nécessaires à notre chatbot et initialisant les variables que nous utiliserons dans notre projet Python.

Le fichier de données est au format JSON, nous avons donc utilisé le package json pour analyser le fichier JSON en Python.

```python
# -*- coding: utf-8 -*-
import nltk
from tensorflow.python.keras.layers import LayerNormalization
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
from tensorflow.keras.models import Sequential
import numpy as np
#from keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
import random
import codecs


words=[] #liste des mots des patterns
classes = [] # va contenir les tags
documents = [] #va contenir liste des sentences d'un pattern suivi de son tag
ignore_words = ['?', '!','-']
#data_file = codecs.open('intents.json',encoding='utf-8').read()
#data_file = open('intents.json').read()
data_file = codecs.open('intents.json',encoding='utf-8').read()
intents = json.loads(data_file)
```

2-3. Prétraiter les données:

Lorsque vous travaillez 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. donc, 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.

```
#tokenize each word
#w=toknizer.tokenize(pattern)
w = nltk.word_tokenize(pattern,language ='french') #  w est une liste des mots
print(w)
words.extend(w) # ajouter chaque w à la liste words ;nous avons utilisé extend car w est une liste: fusionner deux lis
#add documents in the corpus
documents.append((w, intent['tag'])) #ajouter une tuple (w et sa correspondence tag)dans docueents à la fin

# add to our classes list
if intent['tag'] not in classes:
    classes.append(intent['tag'])
```

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.

```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)

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

2-4. Créer des données d'entraînement et de test:

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.

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    #print(output_row)

    training.append([bag, output_row])
    #print(training)
# shuffle our features and turn into np.array
random.shuffle(training)  #Shuffle a list (reorganize the order of the list items): aléatoire,la liste va changer ne trvaille pa
training = np.array(training)
#print(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
#print(train_x)
train_y = list(training[:,1])
print("Training data created")
```

2-5. Construisez le modèle:

Nous avons nos données d'entraînement prêtes, maintenant nous allons construire un réseau neuronal qui a 3 couches. Nous utilisons pour cela l'API séquentielle Keras.

```
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
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")
```

2-6. Prédire la réponse (Interface utilisateur graphique)

Pour prédire les phrases et obtenir une réponse de l'utilisateur, 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 identifient la classe et nous récupérerons 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 :

```
from tensorflow.keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(codecs.open('intents.json',encoding='utf-8').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

Pour prédire la classe, nous devrons fournir des informations. Nous allons donc créer des fonctions qui effectueront un prétraitement du texte puis prédisent la classe.

```
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
```

```
def bow(sentence, words, show_details=True):
    # tokenize the pattern
    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:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))
```

```
def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
```

Après avoir prédit la classe, nous obtiendrons une réponse aléatoire à partir de la liste des intents.

```
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result
```

```
def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

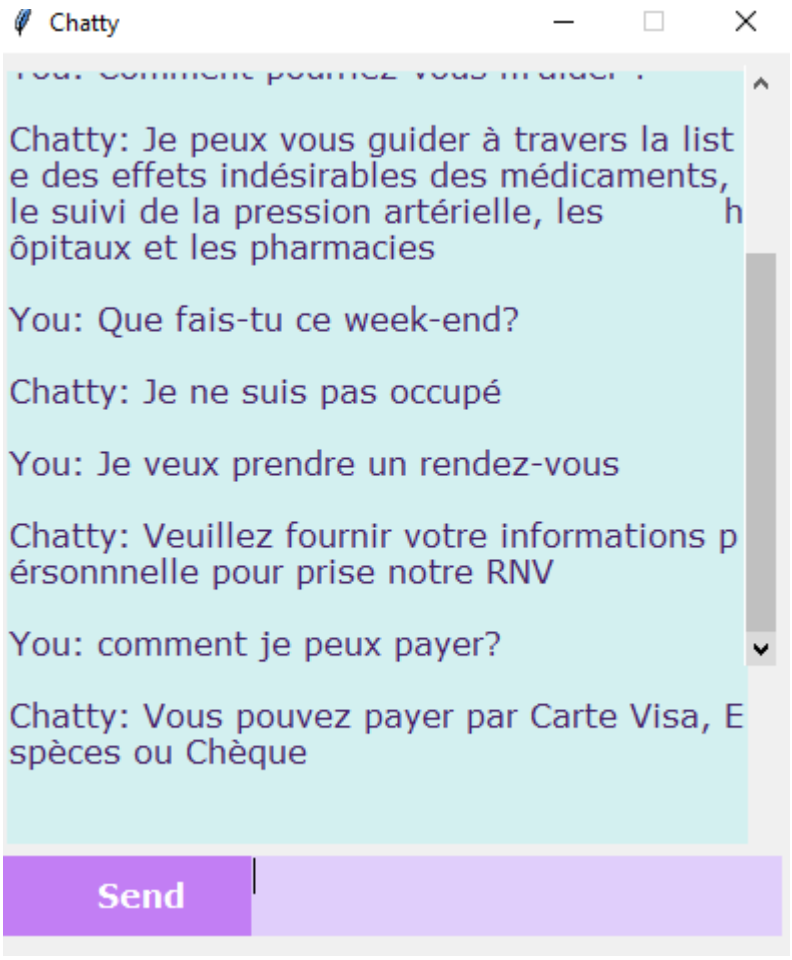
Nous allons maintenant développer une interface utilisateur graphique. Utilisons la bibliothèque Tkinter qui est livrée avec des tonnes de bibliothèques utiles pour l'interface graphique. Nous prendrons le message d'entrée de l'utilisateur, puis utiliserons les fonctions d'assistance que nous

avons créées pour obtenir la réponse du bot et l'afficher sur l'interface graphique. Voici le code source complet de l'interface graphique.

```
#Creating GUI with tkinter
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, "Chatty: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

root = Tk()
root.title("Chatty")
root.geometry("400x500")
root.resizable(width=FALSE, height=FALSE)


#Create Chat window
ChatLog = Text(root, bd=0, bg="white", height="8", width="50", font="Arial",)
```

2-7. Exécutez le chatbot:

Pour exécuter le chatbot il suffit d'exécuter le fichier **Chatbot.ipynb** Le programme ouvrira après quelques secondes l'interface graphique, vous pouvez facilement discuter avec le bot.


Chatty
Chatty: Je peux vous guider à travers la list
e des effets indésirables des médicaments,
le suivi de la pression artérielle, les h
ôpitaux et les pharmacies
You: Que fais-tu ce week-end?
Chatty: Je ne suis pas occupé
You: Je veux prendre un rendez-vous
Chatty: Veuillez fournir votre informations p
érsonnnelle pour prise notre RNV
You: comment je peux payer?
Chatty: Vous pouvez payer par Carte Visa, E
spèces ou Chèque
Send