ROYAUME DU MAROC
UNIVERSITE MOHAMMED V
FACULTE DES SCIENCES RABAT

Master in Data Engineering and Software Development

Question Answering

Edited by:

Mr, AIT EL FILALI BRAHIM

Mr, MARWANE HAREM

Under the supervision of:

Pr. Mahmoudi Abdelhak

PLAN

1) Objectif

2) Structure de projet

3) Nlp concept (steming ,tokenization, bag of words)

4) Save and load model implement the chat

Objectif

Quels sont les avantages de l'utilisation d'un (Question Answering) chatbot ?

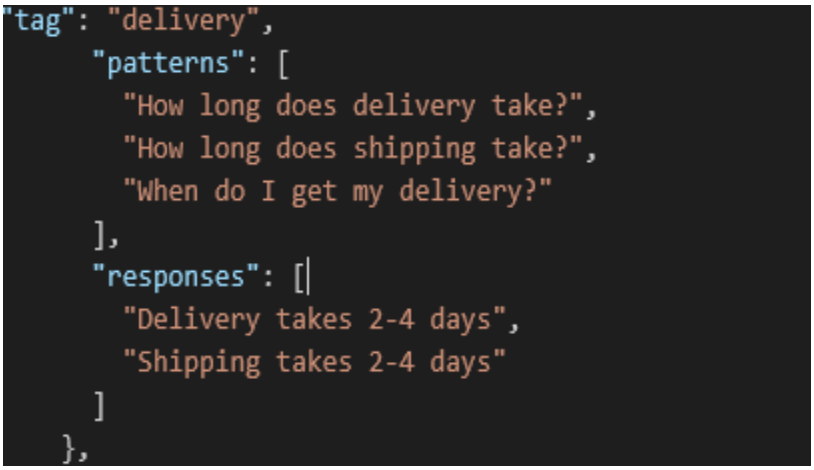
Connaissez-vous les chatbots ? Tout comme les assistants virtuels tels que Siri, Cortana ou encore Google Assistant, les chatbots sont des outils dotés d'Intelligence Artificielle au service de vos clients. Contrairement à ce que l'on pourrait penser, les chatbots ne sont pas une invention récente, mais existent sur le web depuis plusieurs années.

Aujourd'hui, ils sont de plus en plus présents dans nos vies quotidiennes. Les plus grandes entreprises telles que Cdiscount, Air France, Orange ou encore Leroy Merlin, possèdent toutes leur chatbot, souvent représenté sous la forme d'un personnage animé ou plus simplement d'une boîte de dialogue où les clients et les prospects peuvent poser leurs questions.

Notre Mini projet se compose des fichier suivante :

Intents.json : ce fichier contient les tages ainsi que les patternes et aussi les responses, pour notre mini projet on as choisi de travailler sur un chat boot d'un restaurant donc pour ce cas La, Les tags contient les titres globeaux pour les patternes et responses.

For example :



Nltk-util : dans ce classe ont as crier des methodes principale (Tokenzation ,stemming ,bag of words) basé sur la bibliothéque NLTK

Exemple :

```python
def tokenize(sentence):
    """
    split sentence into array of words/tokens
    a token can be a word or punctuation character, or number
    """
    return nltk.word_tokenize(sentence)


def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())
```

Class train :

On a importer le fichier json et après on a crée des tableaux pour gérer les (tags,patterns, respons) pour objectif de réaliser bag_of_words

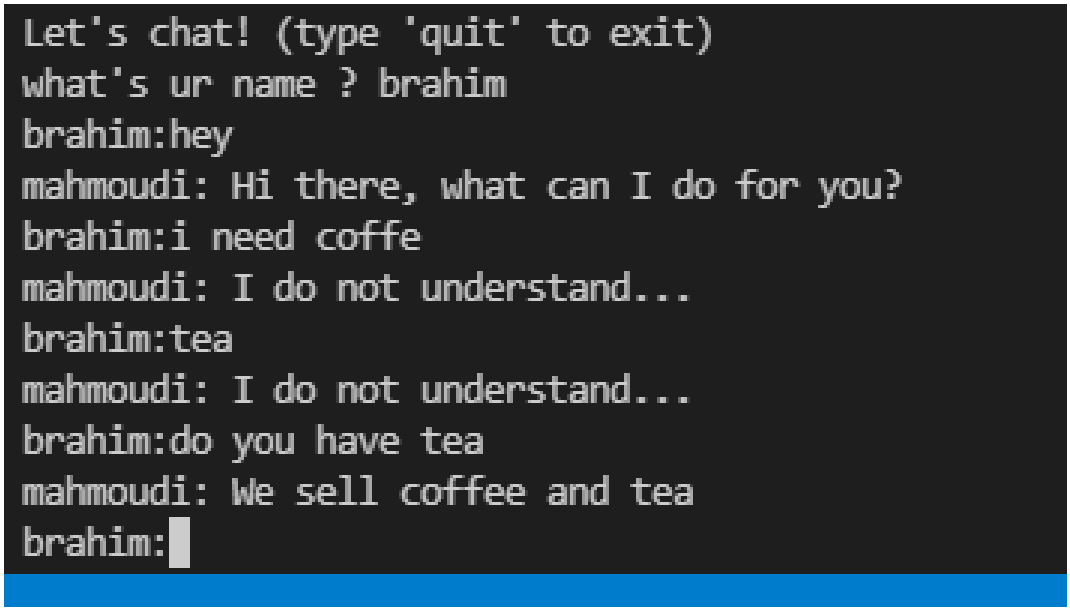
```python
# create training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    # X: bag of words for each pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot
    label = tags.index(tag)
    y_train.append(label)

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

# Hyper-parameters
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)
```

Class chat :

Dans cette classe on a fait appel de toutes les autres classes

```python
bot_name = "mahmoudi"
print("Let's chat! (type 'quit' to exit)")
nom = input("what's ur name ? ")

while True:
    # sentence = "do you use credit cards?"

    sentence = input(nom +":")
    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).to(device)

    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"]:
                print(f"{bot_name}: {random.choice(intent['responses'])}")
```

Demonstration de Notre projet :



Nlp concept (stemming, tokenization, bag of words)

Nettoyage des données

Comme mentionné ci-dessus, le nettoyage des données est une étape basique mais très importante de la PNL. Vous trouverez ci-dessous quelques méthodes de nettoyage des données. Considérons la ligne ci-dessous.

```
all_words = sorted(set(all_words))
```

Faire des minuscules : Ceci est nécessaire pour faire tous les mots en minuscules juste pour maintenir l'uniformité.

Raccourcis : Cela aide à réduire les mots dans leur forme racine. Par exemple:

```
def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())
```

Il existe 2 autres types de tiges en dehors de *PorterStemmer* . Ce sont *Lancaster* Stemming et *Snowball* . Snowball est une amélioration par rapport au stemming de Porter.

Suppression des expressions régulières : L'expression régulière permet d'identifier et de se débarrasser des différents modèles qui ne sont pas requis dans le texte.

B) Tokenisation

C'est l'une des pratiques courantes lorsque l'on travaille sur des données textuelles. Cela permet de diviser une phrase, une phrase ou un paragraphe en petites unités comme des mots ou des termes. Chaque unité est appelée un jeton. Il existe différents types de tokenisation.

On a utilise Tokenisation à l'aide de NLTK

Save and load model implement the chat

Finalement on a exucuté notre projet avec nootbook jupter avec une resultat de 40% donc il a besoin beacoup plus de data training pour augmenter les pérfermonce.