ChatBot project

Team Members :

01. **Romani Nasrat Shawqi "3"**
- 02. **Rawan Abdel-Aziz Ahmed "3"**
- 03 **Reham Mostafa Ali "3"**
- 04. **Sara Reda Moatamed "3"**
- 05. **Rana Hasan Badrawy "3"**
- 06. **Ziad El-Sayed Abdel-Azim "3"**
- .07 **Ahmed Mohamed Ali "1"**

Introduction

-A chatbot is an artificial intelligence (AI) program designed to simulate conversation with human users

-Chatbots use natural language processing (NLP) and machine learning algorithms to understand and respond to user requests, usually through text

- **The purpose of the chatbot project** is to develop an intelligent chatbot using natural language processing (NLP) and neural networks to enhance customer support.

- The chatbot will be designed to assist customers in resolving their queries and issues related to a specific product or service and our project answers short general questions

2. Project Overview

The following are the technologies and tools that will be used in :the chatbot project

Natural Language Processing (NLP): NLP is a subfield of artificial intelligence -1 that deals with the interaction between computers and humans using natural language. It will be used in the project to enable the chatbot to understand and interpret .customer queries

Neural Networks: Neural networks are a set of algorithms that are designed to -2

recognize patterns in data. They will be used to train the chatbot to understand the .customer's intent and provide accurate responses

3-Recurrent Neural Networks (RNNs) *are a type of artificial neural network that are designed to process sequential data, such as time series data, natural language, or music. Unlike traditional feedforward neural networks, RNNs can use their internal state (memory) to process variable-length sequences of inputs. This makes them well-suited for tasks such as language translation, speech recognition, and sentiment analysis.*

4-Python Programming Language: Python is a popular programming language that is widely used for developing artificial intelligence applications. It will be used in the project to develop the chatbot's logic and to integrate it with the company's .customer support system

5-NLTK Library: NLTK is a popular natural language processing library for - Python that provides a set of tools and algorithms for processing human language .data. It will be used to preprocess and analyze text data for training the chatbot

---

Our dataset : it contains random public questions about countries currency , the capital of each of them and a lot of short questions

For example: what is the capital of egypt ?

The chatbot will answer : cairo

```
You:  hi
YOU: hi
Chatbot: Hello
You:  what is the capital of egypt?
YOU: what is the capital of egypt?
Chatbot: Cairo
You:  What is the currency of France?
YOU: What is the currency of France?
Chatbot: Euro (EUR)
You:  How many players in Baseball ?
YOU: How many players in Baseball ?
Chatbot: 9 players on each team (1 pitcher, 1 catcher, 4 infielders, and 3 outfielders)
You:  What is the date today?
YOU: What is the date today?
Chatbot: 10/5/2023
```

-**Preprocessing in NLP refers to the various techniques used to prepare raw text data for machine learning models.**

-**The goal of preprocessing is** to transform raw text data into a format that is easier to analyze and model, while retaining the important information present in the text.

Some common preprocessing techniques used in NLP include:

1-Tokenization: Tokenization refers to the process of breaking a text into individual words or tokens. This is the first step in preprocessing as it provides a way to represent the text in a machine-readable format.

2-Stop word removal: Stop words are commonly occurring words in a language, such as "the," "a," and "is," that do not carry much meaning. Removing stop words from text can help reduce the size of

the dataset and improve the accuracy of the analysis.

3-Stemming and Lemmatization: Stemming and Lemmatization are techniques used to reduce words to their base form. Stemming involves reducing a word to its root form, while lemmatization involves reducing a word to its base form. Both techniques help to reduce the number of unique words in the dataset and improve the accuracy of the analysis.

4-Part-of-speech (POS) tagging: POS tagging involves identifying the part of speech of each word in a text. This can help to identify the context and meaning of the words in the text.

lemmatization

```
[6]: lemmatizer = WordNetLemmatizer()
     x_lemmetized=[]
     for i in range(len(x_tokenized)):
            tokens = x_tokenized[i]
            pos_tags = nltk.pos_tag(tokens)
            lemmas = []
            # print(pos_tags)
            for k in range(len(pos_tags)):
                tag = pos_tags[k][1][0].lower() if pos_tags[k][1][0].lower() in ['a', 'n', 'v'] else 'n'
                lemmas.append(lemmatizer.lemmatize(pos_tags[k][0], tag))
            x_lemmetized.append ( lemmas)

     print(x_lemmetized)
```

```
[['What', 'be', 'the', 'capital', 'of', 'Egypt', '?'], ['What', 'be', 'the', 'capital', 'of', 'Sudan', '?'], ['What', 'be',
'the', 'capital', 'of', 'Zambia', '?'], ['What', 'be', 'the', 'capital', 'of', 'Libya', '?'], ['What', 'be', 'the', 'capita
l', 'of', 'Somalia', '?'], ['What', 'be', 'the', 'capital', 'of', 'Morocco', '?'], ['What', 'be', 'the', 'capital', 'of', 'Ke
nya', '?'], ['What', 'be', 'the', 'capital', 'of', 'Nigeria', '?'], ['What', 'be', 'the', 'capital', 'of', 'Tanzania', '?'],
['How', 'many', 'player', 'in', 'Football', '(', 'soccer', ')', '?'], ['How', 'many', 'player', 'in', 'Volleyball', '?'], ['H
ow', 'many', 'player', 'in', 'Basketball', '?'], ['How', 'many', 'player', 'in', 'Tennis', '?'], ['How', 'many', 'player', 'i
n', 'Baseball', '?'], ['How', 'many', 'player', 'in', 'Hockey', '?'], ['How', 'many', 'player', 'in', 'Rugby', '?'], ['What',
'be', 'the', 'ambulance', 'number', 'in', 'Egypt', '?'], ['What', 'be', 'the', 'Police', 'number', 'in', 'Egypt', '?'], ['Wha
t', 'be', 'the', 'number', 'of', 'Tourist', 'Police', 'in', 'Egypt', '?'], ['What', 'be', 'the', 'Natural', 'Gas', 'Emergenc
y', 'number', 'in', 'Egypt', '?'], ['What', 'be', 'the', 'number', 'of', 'Traffic', 'Police', 'in', 'Egypt', '?'], ['What',
'be', 'the', 'largest', 'organ', 'in', 'the', 'human', 'body', '?'], ['Diabetes', 'affect', 'the', 'body', '’', 's', 'abilit
y', 'to', 'produce', 'which', 'hormone', '?'], ['A', 'cardiologist', 'be', 'a', 'doctor', 'who', 'specialise', 'in', 'conditi
on', 'affect', 'which', 'organ', '?'], ['The', 'fibula', 'and', 'tibia', 'be', 'bone', 'find', 'in', 'which', 'body', 'part',
'?'], ['How', 'many', 'more', 'bone', 'be', 'baby', 'born', 'with', 'than', 'be', 'part', 'of', 'the', 'adult', 'skeleton',
```

```
lemmatizer = WordNetLemmatizer()
x_lemmetized=[]
for i in range(len(x_tokenized)):
       tokens = x_tokenized[i]
       pos_tags = nltk.pos_tag(tokens)
       lemmas = []
       print(pos_tags)
       for k in range(len(pos_tags)):
           tag = pos_tags[k][1][0].lower() if pos_tags[k][1][0].lower() in ['a', 'n', 'v'] else 'n'
           lemmas.append(lemmatizer.lemmatize(pos_tags[k][0], tag))
       x_lemmetized.append ( lemmas)

print(x_lemmetized)
```

```
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Sudan', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Zambia', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Libya', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Somalia', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Morocco', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Kenya', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Nigeria', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('capital', 'NN'), ('of', 'IN'), ('Tanzania', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Football', 'NNP'), ('(', '('), ('soccer', 'NN'), (')', ')'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Volleyball', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Basketball', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Tennis', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Baseball', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Hockey', 'NNP'), ('?', '.')]
[('How', 'WRB'), ('many', 'JJ'), ('players', 'NNS'), ('in', 'IN'), ('Rugby', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('ambulance', 'NN'), ('number', 'NN'), ('in', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('Police', 'NNP'), ('number', 'NN'), ('in', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('number', 'NN'), ('of', 'IN'), ('Tourist', 'NNP'), ('Police', 'NNP'), ('in', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('Natural', 'NNP'), ('Gas', 'NNP'), ('Emergency', 'NNP'), ('number', 'NN'), ('in', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('number', 'NN'), ('of', 'IN'), ('Traffic', 'NNP'), ('Police', 'NNP'), ('in', 'IN'), ('Egypt', 'NNP'), ('?', '.')]
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('largest', 'JJS'), ('organ', 'NN'), ('in', 'IN'), ('the', 'DT'), ('human', 'JJ'), ('body', 'NN'), ('?', '.')]
```

Preprocessing in NLP is a crucial step in building accurate and effective machine learning models. By transforming raw text data into a machine-readable format, preprocessing enables machine learning models to effectively analyze and model text data, making it a valuable tool in a range of NLP applications.

In the context of a chatbot

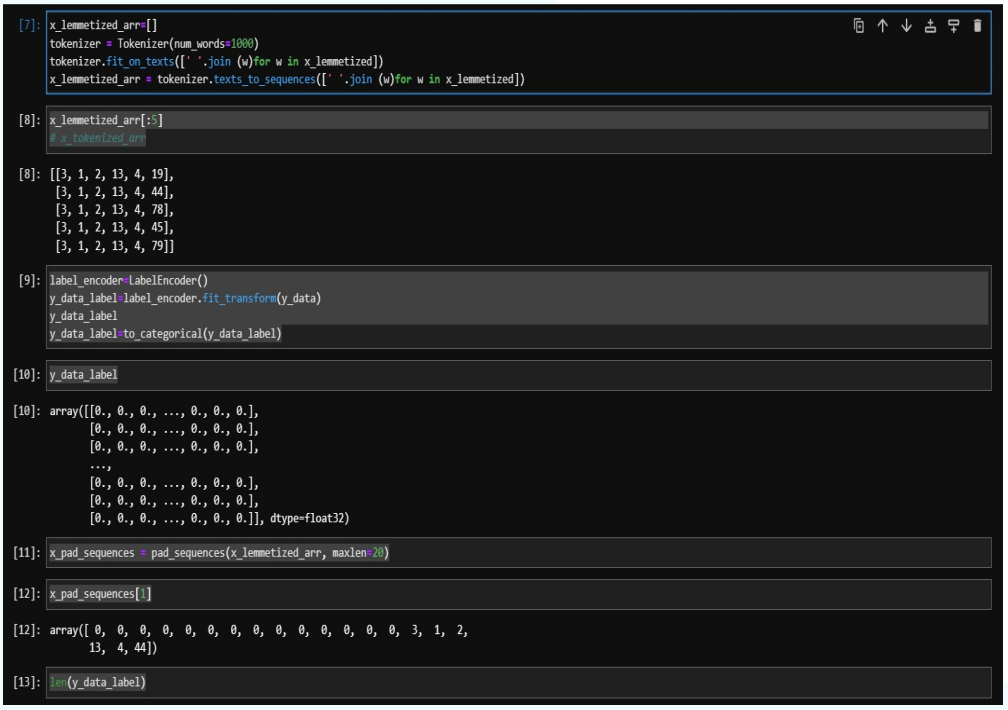
an encoder is used to convert the input text into a fixed-length vector representation that can be used as input to a machine learning model, such as a neural network. This vector representation is called an encoding.

Padding sequences is a common technique used in natural language processing, including in chatbot development. It involves **adding zeros or other padding tokens to the end of input sequences so that all sequences are of equal length**.

-In the context of chatbots, padding sequences is often used to ensure that all input texts have the same length, which allows for more efficient processing by the neural network. Without padding, inputs of varying lengths would need to be processed separately, which can be computationally expensive and slow down training.

The reason for using an encoder is that neural networks typically require a **fixed-size input**, whereas natural language inputs can be of varying length. By using an encoder to convert the input text into a fixed-size encoding, the chatbot model can process the input in a consistent and efficient manner.

One popular type of encoder used in chatbots is the **Recurrent Neural Network (RNN) encoder**, which processes the input text word-by-word or character-by-character, and uses a recurrent layer to capture the context and dependencies between the words or characters.



```
[7]: x_lemmetized_arr=[]
     tokenizer = Tokenizer(num_words=1000)
     tokenizer.fit_on_texts([' '.join (w)for w in x_lemmetized])
     x_lemmetized_arr = tokenizer.texts_to_sequences([' '.join (w)for w in x_lemmetized])

[8]: x_lemmetized_arr[:5]
     # x_tokenized_arr

[8]: [[3, 1, 2, 13, 4, 19],
      [3, 1, 2, 13, 4, 44],
      [3, 1, 2, 13, 4, 78],
      [3, 1, 2, 13, 4, 45],
      [3, 1, 2, 13, 4, 79]]

[9]: label_encoder=LabelEncoder()
     y_data_label=label_encoder.fit_transform(y_data)
     y_data_label
     y_data_label=to_categorical(y_data_label)

[10]: y_data_label

[10]: array([[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)

[11]: x_pad_sequences = pad_sequences(x_lemmetized_arr, maxlen=20)

[12]: x_pad_sequences[1]

[12]: array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,  1,  2,
             13,  4, 44])

[13]: len(y_data_label)
```

Rnn model

Input layer: The input layer receives the input sequence, usually a sequence of words or tokens. In NLP tasks, these inputs are usually represented as vectors, such as word embeddings.
Recurrent layer: The recurrent layer is where the RNN captures the sequential information in the input sequence. It uses recurrent connections to pass information from one time step to the next, allowing the model to remember previous inputs and context.
Activation function: The activation function is applied to the output of the recurrent layer to introduce non-linearity into the model. Common activation functions used in RNNs include the sigmoid function and the

hyperbolic tangent function.
Output layer: The output layer produces the final output of the model, usually a prediction or classification. In NLP tasks, the output layer can be used for tasks such as language modeling, sentiment analysis, and machine translation

```
kerasModel = keras.models.Sequential([
# ----------------------
    keras.layers.Embedding(input_dim=1000, output_dim=32,input_length=20),
        keras.layers.Flatten(),
        # keras.layers.Dense(units=32, activation='relu'),

        # keras.layers.Dense(units=128, input_shape=(None,len(lenMAX)), activation='relu'),
        # keras.layers.Dense(units=128, activation='relu'),

        # keras.layers.Dropout(rate=0.5) ,
        # keras.layers.Dense(units=256, activation='relu'),
        #
        keras.layers.Dense(units=128, activation='relu'),
        keras.layers.Dense(units=64, activation='relu'),
        keras.layers.Dense(units=32, activation='relu'),


        # keras.layers.Dense(3512,activation='softmax') ,
        keras.layers.Dense(124,activation='softmax') ,



        ])
```

Training the Model: Once the model is built, it can be trained using the training data. During training, the model is presented with input sequences and corresponding output sequences, and adjusts its weights and biases to minimize the loss function. This process is repeated for a specified number of epochs or until the model reaches a certain level of accuracy.

```
[15]: kerasModel.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
      # model.summary()

[24]: kerasModel.fit(x_pad_sequences,y_data_label,epochs=40)

      Epoch 1/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1690 - accuracy: 0.9618
      Epoch 2/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1585 - accuracy: 0.9542
      Epoch 3/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1364 - accuracy: 0.9542
      Epoch 4/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1255 - accuracy: 0.9542
      Epoch 5/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1124 - accuracy: 0.9542
      Epoch 6/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1082 - accuracy: 0.9618
      Epoch 7/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1145 - accuracy: 0.9695
      Epoch 8/40
      5/5 [==============================] - 0s 2ms/step - loss: 0.1020 - accuracy: 0.9542
```

Testing the Model: After the model is trained, it can be tested using the testing data to evaluate its performance. This involves presenting the model with input sequences and comparing its predicted output sequences to the actual output sequences. Metrics such as accuracy, precision, recall, and F1 score can be used to evaluate the model's performance.

```
model=load_model('modelChatBot222.h5')

val_loss, val_acc = model.evaluate(x_pad_sequences, y_data_label)
print(val_loss)
print(val_acc)

5/5 [==============================] - 0s 2ms/step - loss: 0.0469 - accuracy: 0.9695
0.046944353729486465
0.9694656729698181
```

-

This cell to interact with chatbot

The user types his question and the chatbot predicts the answer

[21]:
```python
def preprocess_input(input_text):
    input_text = input_text.lower()
    input_text = nltk.word_tokenize(input_text)
    input_text = [lemmatizer.lemmatize(word) for word in input_text]
    input_text = tokenizer.texts_to_sequences([input_text])
    input_text = tf.keras.preprocessing.sequence.pad_sequences(input_text, maxlen=20)
    return input_text




while True:
    user_input = input("You: ")
    preprocessed_input = preprocess_input(user_input)
    prediction = model.predict(preprocessed_input)
    predicted_label = label_encoder.inverse_transform([np.argmax(prediction)])


    print("YOU: " + user_input )
    # print(predicted_label[0])
    print("Chatbot: " + predicted_label[0])
```