Project Report

On

ACP FRENCH TRANSLATOR THROUGH MACHINE LEARNING MODEL

Submitted by

Y Asha-R171111
P Naga Hari Chandana R171028
U Pavithra-R170085

Under the guidance of

P Ravi Kumar
M.E(IISc Banglore),Asst Proffesor

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

R.K.Valley, Kadapa, Andra Pradesh.

Rajiv Gandhi University of Knowledge Technologies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled "**COVID TESTING MANAGEMENT SYSTEM**" is a bonafied project work submitted by P Naga Hari Chanadana U Pavithra Y Asha in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2021-2022 carried out the work under the supervision

GUIDE
P RAVI KUMAR

HEAD OF THE DEPARTMENT
P HARINADHA

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and Whose constant guidance and encouragement crown all the efforts success.

We are extremely grateful to our respected Director, Prof. K. SANDHYA RANI for Fostering an excellent academic climate in our institution.

We also express my sincere gratitude to our respected Head of the Department Mr P HARINADHA for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

We would like to convey thanks to our guide at college Mr P RAVI KUMAR for his Guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

Our sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

INDEX

ABSTRACT

A language translator is an app which converts text we entered in one language to another Language . We need language translator for better communication where there is no common mode of language . It makes it easy for people to translate text in one language to text in another language.

In this project we tried to develop a ENGLISH-FRENCH Language Translator.Our main intention is to use this translator instead classic dictonaries for language translation. This system is designed to overcome all challenges related to the classic dictonary translation and online translator.

The system is an offiline which will work without internet, so we do not have any delay and network issues while translating.It gives results faster than any online translator and using our system more reliable than using dictonaries.

INTRODUCTION

In recent many language translators based on machine learning have been introduced in market in order to translate one language into another desired language. Although there are many translators available there are very few translators for english to
french language, also not all systems are accurable and available only on online.

The purpose of this project is to develop accurate and an offline english to french language translator.In this machine learning project, we developed a Language Translator using a many-to-many encoder-decoder sequence model. We trained our model using LSTM which will convert English to French language where English will be input text and French will be the target text. For this, we will be using English-French dataset.

In this system we have only user module.

User:

- User can enter the english text that needs to be translated to french, the system will display the translated french text.

PURPOSE

The main purpose of ACP French Translator to provide a platform where users can translate their sentences from english to french.With the help of this project we use of technology to translate a language into another language with out internet.Another purpose of this project is to replace the classic dictonaries for language translation.

SCOPE

In earlier days we have to invest large amount of time to translate our language using the dictionary.As Technology is growing rapidly we are also moving to a technical world where everything we want to be automatic and reduce our work.This project makes english to french language translation process easy and reduce the burden of the users.

This access friendly software provides quick and effective services which helps user to reduce thier surfing time for translation.

Advantages:

- Easy, user friendly GUI.
- Very accurate results.
- The system translates English to French language without internet(offline).

Disadvantages:

- The system only translates from English to French language only.

- It reduces the sales of dictionaries.

REQUIREMENT SEPCIFCATION

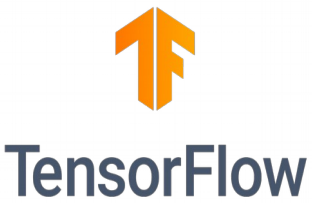
Hardware Configuration:

Ram	4GB
Hard Disk	512GB
Processor	2Gz

Software Requirement:

	Requirement	**Version**
Language	Python	3.8.5 and above
Module	Pickle	4.0 and above
Module	Tenserflow	2.2 and above
Module	Sklearn	0.24.2 and above
Additional	Numpy	1.19.5 and above
Operating System	Windows and Ubuntu	10 and above,& 16.04 and above
GUI	Tkinker	

TENSER FLOW:

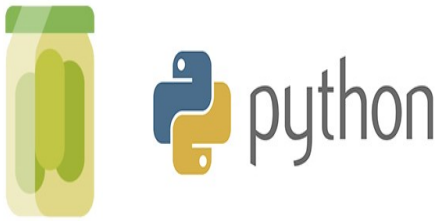


TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. See the TensorFlow documentation complete details on the broader TensorFlow System.

TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you used a high-level API named tf.keras to define and train machine learning

models and to make predictions. tf.keras is the TensorFlow variant of the open-source KerasAPI.

PICKLE:



If you need to save complex Python data structures, and only expect to load it in the future into Python, then you can consider using Python's pickle module.

The pickle module is used for storing Python object structures into a file (and retrieving it back into memory at a later time).

For example, you may use it to save your Machine Learning model that you have been spending the whole week training.

You pickle your Python objects onto the disk as a **binary file**(serialising), and you unpickle them from the disk into memory (deserialising).

You can pickle integers, floats, booleans, strings, tuples, lists, sets, dictionaries (that contain objects that can be pickled), top-level classes. No pickled gherkins, sorry!

Health warnings!

1.Pickle is specific to Python. It is not recommended if you expect to share your data across different programming languages

2.Make sure you use the same Python version. It is not guaranteed that pickle will work with different versions of Python

3.Do not unpickle data from untrusted sources as you may execute malicious code inside the file when unpickling.

SKLEARN:

•The scikit-learn library is one of the most popular platforms for everyday machine learning and data science. The reason is because it is built upon Python, a fully featured programming language.

•But how do you get started with machine learning with scikit-learn.

•Kevin Markham is a data science trainer who created a series of 9 videos that show you exactly how to get started in machine learning with scikit-learn.

•In this post you will discover this series of videos and exactly what is covered, step-by-step to help you decide if the material will be useful to you.

Numpy:

What is NumPy?

1.NumPy is a Python library used for working with arrays.

2.It also has functions for working in domain of linear algebra, fourier transform, and matrices.

3.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely

4.NumPy stands for Numerical Python.

Why Use NumPy?

1.In Python we have lists that serve the purpose of arrays, but they are slow to process.

2.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

3.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

4.Arrays are very frequently used in data science, where speed and resources are very important.

ANALYSIS & DESIGN

Today also we have to use dictionaries for language translation or any online translator. If we use dictionaries which is old method need lots of time and concentration.Instead if we use online translators in case of any network issues it won't work. So with the help of this project we are making an ofline translator which faster and more reliable than existing methods.This project reduces the translation easy and it is available offline.At the same time it is very accurate.

Disadvantages of present system:

- It need more manual effort if we use dictionaries.
- Time consuming.
- Not user friendly.
- If we use online translator network issues will occur.

Method used:

Machine Learning: Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning " in 1959 while at IBM. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed ". However, there is no universally accepted definition for machine learning. Different authors define the term differently. We give below two more definitions.

•Machine learning is programming computers to optimize a performance criterion using example data or past experience . We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.

The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

Project file structure:

- eng-french.txt: This is dataset for our project, contains English text and corresponding french texts.
- langTraining.py: This file is used to create and train the model.
- training_data.pkl: This file contains lists of characters, text of input, and target data in a binary format.
- s2s: This directory contains optimizer, metrics, and weights of our trained model.
- LangTransGui.py: Gui file of our project where we load the trained model and predict output for given text.

Coding

Model Used:

Recurrent Neural network Model.

Import Libraries and initialize variables:

First, we need to import all the libraries and need intialize the variables.

#code:

```
from tensorflow.keras.models import Model
from tensorflow.keras import models
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import Input,LSTM,Dense
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
import pickle
```

Intialize Variables:

#code:

```
input_texts=[]
target_texts=[]
input_characters=set()
target_characters=set()
```

Parse the data set file:

Now we need to parse the datasetfile by using “Teacher Forcing Algorithm”.

Teacher Forcing Algorithm:

Teacher Forcing is a method for a quickly and efficiently training reccurent neural network models that use the ground truth from a prior time step as input.

For example, we want to predict the next word from the sequence ‘Python is a programming language’. So we will put ‘\t’ at the start and ‘\n’ at the end of the text(sequence) to tell the model that this is the starting and ending of the text which becomes ‘\tPython is a programming language\n’. So we feed ‘\t’ to the model at the first timestep and so on for every timestep.

(x)	**(y^)**
\t	Python
\tPython,	is
\tPython is,	a

As you can see after feeding input(x) it is predicting the next word(y^), it will do the same till it reaches ‘\n’ which is our end of the sentence.

Data in file:

Go.	Va !
Run!	Cours !
Run!	Courez !

Wow! Ça alors !
Fire! Au feu !
Help! À l'aide !

read dataset file:

```
with open('eng-french.txt','r',encoding='utf-8') as f:
rows=f.read().split('\n')
```

read first 10,000 rows from dataset :

```
for row in rows[:10000]:
split input and target by '\t'=tab:
input_text,target_text = row.split('\t')
```

add '\t' at start and '\n' at end of text:

```
target_text='\t' + target_text + '\n'
input_texts.append(input_text.lower())
target_texts.append(target_text.lower())
```

#split character from text and add in respective sets:

```
input_characters.update(list(input_text.lower()))
target_characters.update(list(target_text.lower()))
```

We used the same procedure and separate the text from rows and characters. Also, get the maximum length of encoder as well as decoder sequence. Repeat the same for target text as well

#Code:

```
#sort input and target characters :
input_characters = sorted(list(input_characters))
target_characters = sorted(list(target_characters))
#get the total length of input and target characters:
num_en_chars = len(input_characters)
num_dec_chars = len(target_characters)
#get the maximum length of input and target text:
max_input_length = max([len(i) for i in input_texts])
max_target_length = max([len(i) for i in target_texts])
print("number of encoder characters : ",num_en_chars)
print("number of decoder characters : ",num_dec_chars)
print("maximum input length : ",max_input_length)
print("maximum target length : ",max_target_length)
```

One Hot Encoding (Vectorization):

Models cannot work directly on the categorical data. For this, we require one hot encoding process.

One-hot encoding deals with the data in binary format so we encode the categorical data in binary format.

One-hot means that we can only make an index of data 1 (true) if it is present in the vector or else 0 (false). So every data has its unique representation in vector format.

For example, if we have an array of data like : [“python”,”java”,”c++”] then the one hot encoding representation of this array will be :

[[1 , 0 , 0]
[0 , 1 , 0]
[0 , 0 , 1]]

In our project after separating characters from input and target text we used a one-hot encoding process. We will fit characters and transform the texts accordingly. So if the character from input text is present in the sets of characters then it will put 1 and 0 otherwise.

```
def bagofcharacters(input_texts,target_texts):
#initialize encoder , decoder input and target data.
en_in_data=[] ; dec_in_data=[] ; dec_tr_data=[]
#padding variable with first character as 1 as rest all 0.
pad_en=[1]+[0]*(len(input_characters)-1)
pad_dec=[0]*(len(target_characters)) ; pad_dec[2]=1
#countvectorizer for one hot encoding as we want to tokenize character so
#analyzer is true and None the stopwords action.
cv=CountVectorizer(binary=True,tokenizer=lambda txt:
txt.split(),stop_words=None,analyzer='char')
for i,(input_t,target_t) in enumerate(zip(input_texts,target_texts)):
#fit the input characters into the CountVectorizer function
cv_inp= cv.fit(input_characters)
#transform the input text from the help of CountVectorizer fit.
#it character present than put 1 and 0 otherwise.
en_in_data.append(cv_inp.transform(list(input_t)).toarray().tolist())
cv_tar= cv.fit(target_characters)
dec_in_data.append(cv_tar.transform(list(target_t)).toarray().tolist())
#decoder target will be one timestep ahead because it will not consider
#the first character i.e. '\t'.
dec_tr_data.append(cv_tar.transform(list(target_t)[1:]).toarray().tolist())
```

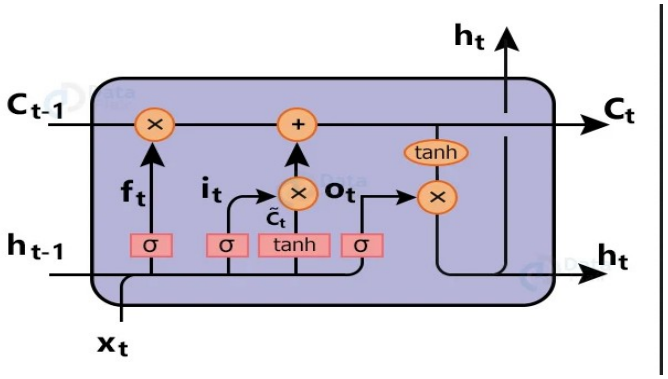
Code:

```
#add padding variable if the length of the input or target text is smaller
#than their respective maximum input or target length.
if len(input_t) < max_input_length:
for _ in range(max_input_length-len(input_t)):
en_in_data[i].append(pad_en)
if len(target_t) < max_target_length:
for _ in range(max_target_length-len(target_t)):
dec_in_data[i].append(pad_dec)
if (len(target_t)-1) < max_target_length:
for _ in range(max_target_length-len(target_t)+1):
dec_tr_data[i].append(pad_dec)
#convert list to numpy array with data type float32
en_in_data=np.array(en_in_data,dtype="float32")
dec_in_data=np.array(dec_in_data,dtype="float32")
dec_tr_data=np.array(dec_tr_data,dtype="float32")
return en_in_data,dec_in_data,dec_tr_data
```

Also, we need the same length of input data throughout so we will be adding an extra array of 0’s to make all input text the same vector length. Repeat the same procedure for target data also.

To Build the Training Model:

In this project we used LSTM to train our machine learning model.



LSTM (Long Short Term Memory) network: LSTM is a type of RNN (Recurrent Neural Network) that solves scenarios where RNN is failed.

Long-Term Dependency: In RNN, networks have the data of previous output in memory for a short period of time because of this they are unaware about the actual context of the sentence over a long period of time. This raised the issue of long-term dependency.

Cell Memory state (ct): Cell state is actually what makes LSTM a unique network. Cell state holds the memory for over a long period of time. Data can be removed or added in cell state depending upon the layer requirements.

Hidden state (ht): hidden state is basically output of the previous block. We decide what to do with the memory looking at the previous hidden state output and current input. And also we don't want output after every timestep until we reach the last input of our sequence.

Forget Gate (ft): Forget Gate is used to check what data we want to neglect away from the cell state. This is done using a sigmoid layer. This Gate looks at hidden output from previous time steps and current input, after that it outputs number 0 which means neglect the data, or 1 means keep the data.

Input Gate: We want to check what new information we are going to store in the cell memory state (ct). So data will pass through the sigmoid function which decides which values to update (it). Next a tanh function creates a vector of new candidates (čt) for our state.

Output Gate: We want to pass the output to the next layer. As the output is dependent upon the cell state, we will be using sigmoid which decides what parts of the cell state we are going to output. Then we will apply tanh and multiply them with sigmoid value of output (ot) so that we only output values that are required.

So we will pass cell state and hidden state value output to another block of the LSTM layer as an input. And this cell and hidden output are passed to the next block of LSTM layers where the same steps repeat till we reach the right prediction.

Encoder: For the language translation machine learning model we will be creating keras Input objects whose shape will be equal to the total number of input characters. We used RNN's LSTM model and only the return state will be True because we only want value from hidden and cell state so we will discard encoder output and only keep the states.

Code:

```
#create input object of total number of encoder characters
en_inputs = Input(shape=(None, num_en_chars))
#create LSTM with the hidden dimension of 256
#return state=True as we don't want output sequence.
encoder = LSTM(256, return_state=True)
#discard encoder output and store hidden and cell state.
en_outputs, state_h, state_c = encoder(en_inputs)
en_states = [state_h, state_c]
```

Decoder: In decoder, our Input object shape will be equal to the total number of target characters. The LSTM model with the return state and return sequence will be True as we need a model to return full output sequence(text) as well as states. We will be using a softmax activation function and also a Dense layer for our output.

Code:

```
#create input object of total number of decoder characters
dec_inputs = Input(shape=(None, num_dec_chars))
#create LSTM with the hidden dimension of 256
#return state and return sequences as we want output sequence.
dec_lstm = LSTM(256, return_sequences=True, return_state=True)
#initialize the decoder model with the states on encoder.
dec_outputs, _, _ = dec_lstm(dec_inputs, initial_state=en_states)
#Output layer with shape of total number of decoder characters
dec_dense = Dense(num_dec_chars, activation="softmax")
dec_outputs = dec_dense(dec_outputs)
```

Train the model

To train the model we will fit '(encoder input and decoder input)' which will turn into ('decoder target data') using 'Adam' optimizer with a validation split of 0.2 and provide an epoch of 200 in a batch size of 64. Also, we will store all required variables in a binary or bytes stream like object format file using the 'pickle' module

Code:

```
#create Model and store all variables
model = Model([en_inputs, dec_inputs], dec_outputs)
pickle.dump({'input_characters':input_characters,'target_characters':target_characters,
'max_input_length':max_input_length, 'max_target_length':max_target_length,
'num_en_chars':num_en_chars, 'num_dec_chars':num_dec_chars}, open("training_data.pkl", "wb"))
```

```
#load the data and train the model
en_in_data,dec_in_data,dec_tr_data = bagofcharacters(input_texts,target_texts)
model.compile(
optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"]
)
model.fit(
[en_in_data, dec_in_data],
dec_tr_data,
batch_size=64,
epochs=200,
validation_split=0.2,
)
# Save model
model.save("s2s")
#summary and model plot
model.summary()
plot_model(model, to_file='Model_plot.png', show_shapes=True, show_layer_names=True)
```

Create GUI for our prediction:

Now we will create a GUI for language translation machine learning project using the “tkinter” module to get input and display the decoded (translated) french text on the display window.

Initialize the window and load all the modules:

Code:

```
from tkinter import *
import pickle
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from tensorflow.keras.models import Model
from tensorflow.keras import models
from tensorflow.keras.layers import Input,LSTM,Dense
BG_GRAY="#ABB2B9"
BG_COLOR="#000"
TEXT_COLOR="#FFF"
FONT="Melvetica 14"
FONT_BOLD="Melvetica 13 bold"
cv=CountVectorizer(binary=True,tokenizer=lambda txt: txt.split(),stop_words=None,analyzer='char')
class LangTRans:
def __init__(self):
#initialize tkinter window and load the file
self.window=Tk()
self.main_window()
self.datafile()
```

Load all the variables from the datafile using the pickle module:

```
def datafile(self):
#get all datas from datafile and load the model.
```

```
datafile = pickle.load(open("training_data.pkl","rb"))
self.input_characters = datafile['input_characters']
self.target_characters = datafile['target_characters']
self.max_input_length = datafile['max_input_length']
self.max_target_length = datafile['max_target_length']
self.num_en_chars = datafile['num_en_chars']
self.num_dec_chars = datafile['num_dec_chars']
self.loadmodel()
```

Create the main window for language translation. Also create scrollbar, text widget for the GUI:

```
def main_window(self):
#add title to window and configure it
self.window.title("Language Translator")
self.window.resizable(width=False,height=False)
self.window.configure(width=520,height=520,bg=BG_COLOR)
head_label=Label(self.window,bg=BG_COLOR, fg=TEXT_COLOR, text="Welcome to ACP FRENCH
TRANSLATOR",font=FONT_BOLD, pady=10)
head_label.place(relwidth=1)
line = Label(self.window,width=450,bg=BG_COLOR)
line.place(relwidth=1,rely=0.07,relheight=0.012)
#create text widget where input and output will be displayed
self.text_widget=Text(self.window,width=20,height=2,bg="#fff",fg="#000",
font=FONT,padx=5,pady=5)
self.text_widget.place(relheight=0.745,relwidth=1,rely=0.08)
self.text_widget.configure(cursor="arrow",state=DISABLED)
#create scrollbar
scrollbar=Scrollbar(self.text_widget)
scrollbar.place(relheight=1,relx=0.974)
scrollbar.configure(command=self.text_widget.yview)
#create bottom label where text widget will placed
bottom_label=Label(self.window,bg=BG_GRAY,height=80)
bottom_label.place(relwidth=1,rely=0.825)
#this is for user to put english text
self.msg_entry=Entry(bottom_label,bg="#2C3E50", fg=TEXT_COLOR,font=FONT)
self.msg_entry.place(relwidth=0.788,relheight=0.06,rely=0.008,relx=0.008)
self.msg_entry.focus()
self.msg_entry.bind("<Return>",self.on_enter)
#send button which will call on_enter function to send the text
send_button=Button(bottom_label,text="Send",font=FONT_BOLD,
width=8,bg="#fff",command=lambda: self.on_enter(None))
send_button.place(relx=0.80,rely=0.008,relheight=0.06,relwidth=0.20)
```

Inference (sampling) model and prediction

Load the saved model and construct encoder and decoder. We will get the inputs from the saved model and LSTM to get the hidden and cell state of the encoder which is required to create the encoder model.

Code:

```
def loadmodel(self):
#Inference model
#load the model
model = models.load_model("s2s")
```

```
#construct encoder model from the output of second layer
#discard the encoder output and store only states.
enc_outputs, state_h_enc, state_c_enc = model.layers[2].output
#add input object and state from the layer.
self.en_model = Model(model.input[0], [state_h_enc, state_c_enc])
```

For the decoder, we will take the second input and create an input object for hidden as well for cell state of shape (256,) which is latent(hidden) dimension of layer. Also, we will run one step of the decoder with this initial state and a start of text character after that our output will be the next character of the text.

We used reverse lookup to get characters from the index of the ‘input_text’ variable.

Code:

```
#create Input object for hidden and cell state for decoder
#shape of layer with hidden or latent dimension
dec_state_input_h = Input(shape=(256,), name="input_3")
dec_state_input_c = Input(shape=(256,), name="input_4")
dec_states_inputs = [dec_state_input_h, dec_state_input_c]
#add input from the encoder output and initialize with states.
dec_lstm = model.layers[3]
dec_outputs, state_h_dec, state_c_dec = dec_lstm(
model.input[1], initial_state=dec_states_inputs
)
dec_states = [state_h_dec, state_c_dec]
dec_dense = model.layers[4]
dec_outputs = dec_dense(dec_outputs)
#create Model with the input of decoder state input and encoder input
#and decoder output with the decoder states.
self.dec_model = Model(
[model.input[1]] + dec_states_inputs, [dec_outputs] + dec_states
)
```

Encode the input sequence as state vectors. Create an empty array of the target sequence of length 1 and generate the start character i.e ‘\t’ in our case of every pair to be 1. Use this state value along with the input sequence to predict the output index. Use reverse character index to get the character from output index and append to the decoded sequence.

Code:

```
def decode_sequence(self,input_seq):
#create a dictionary with a key as index and value as characters.
reverse_target_char_index = dict(enumerate(self.target_characters))
#get the states from the user input sequence
states_value = self.en_model.predict(input_seq)
#fit target characters and
#initialize every first character to be 1 which is '\t'.
```

```
#Generate empty target sequence of length 1.
co=cv.fit(self.target_characters)
target_seq=np.array([co.transform(list("\t")).toarray().tolist()],dtype="float32")
#if the iteration reaches the end of text than it will be stop the it
stop_condition = False
#append every predicted character in decoded sentence
decoded_sentence = ""
while not stop_condition:
#get predicted output and discard hidden and cell state.
output_chars, h, c = self.dec_model.predict([target_seq] + states_value)
#get the index and from the dictionary get the character.
char_index = np.argmax(output_chars[0, -1, :])
text_char = reverse_target_char_index[char_index]
decoded_sentence += text_char
```

For every index, put 1 to that index of our target array. So for the next iteration, our target sequence will be having a vector of the previous character. Iterate until our character is equal to the last character or max length of the target text.

Code:

```
# Exit condition: either hit max length
# or find a stop character.
if text_char == "\n" or len(decoded_sentence) > self.max_target_length:
stop_condition = True
#update target sequence to the current character index.
target_seq = np.zeros((1, 1, self.num_dec_chars))
target_seq[0, 0, char_index] = 1.0
states_value = [h, c]
#return the decoded sentence
return decoded_sentence
```

Get the input (English) text from the user and pass it to a bag of characters for a one-hot encoding process. After that pass the encoded vector into ‘decode_sequence()’ for the decoded(french) text.

Code

```
def bagofcharacters(self,input_t):
cv=CountVectorizer(binary=True,tokenizer=lambda txt:
txt.split(),stop_words=None,analyzer='char')
en_in_data=[] ; pad_en=[1]+[0]*(len(self.input_characters)-1)
cv_inp= cv.fit(self.input_characters)
en_in_data.append(cv_inp.transform(list(input_t)).toarray().tolist())
if len(input_t)< self.max_input_length:
for _ in range(self.max_input_length-len(input_t)):
en_in_data[0].append(pad_en)
return np.array(en_in_data,dtype="float32")
def decoded_output(self,msg,sender):
self.text_widget.configure(state=NORMAL)
```

```
en_in_data = self.bagofcharacters(msg.lower()+".")
self.text_widget.insert(END,str(sender)+" :
"+self.decode_sequence(en_in_data)
+"\n\n")
self.text_widget.configure(state=DISABLED)
self.text_widget.see(END)
def my_msg(self,msg,sender):
if not msg:
return
self.msg_entry.delete(0,END)
self.text_widget.configure(state=NORMAL)
self.text_widget.insert(END,str(sender)+" : "+str(msg)+"\n")
self.text_widget.configure(state=DISABLED)
#runwindow
def run(self):
self.window.mainloop()
# run the file
if __name__=="__main__":
LT = LangTRans()
LT.run()
```

Run Language Translation Code File

In order to run the language translator app, we used these two main files langTraining.py and LangTransGui.py.

First, we trained the model by using the following command in the terminal.

“python langTraining.py”

After entering the command we get as follows:

```
Epoch 1/200
125/125 [==============================] - 4s 16ms/step - loss: 1.2860 - accuracy: 0.7175 - val_loss: 1.1168 - val_accuracy: 0.6876
Epoch 2/200
125/125 [==============================] - 1s 11ms/step - loss: 0.8887 - accuracy: 0.7539 - val_loss: 0.9507 - val_accuracy: 0.7399
Epoch 3/200
125/125 [==============================] - 1s 11ms/step - loss: 0.7759 - accuracy: 0.7860 - val_loss: 0.8393 - val_accuracy: 0.7586
Epoch 4/200
125/125 [==============================] - 1s 11ms/step - loss: 0.6664 - accuracy: 0.8081 - val_loss: 0.7686 - val_accuracy: 0.7754
Epoch 5/200
125/125 [==============================] - 1s 11ms/step - loss: 0.6173 - accuracy: 0.8195 - val_loss: 0.7266 - val_accuracy: 0.7846
Epoch 6/200
125/125 [==============================] - 1s 11ms/step - loss: 0.5845 - accuracy: 0.8281 - val_loss: 0.6940 - val_accuracy: 0.7948
Epoch 7/200
125/125 [==============================] - 1s 11ms/step - loss: 0.5582 - accuracy: 0.8354 - val_loss: 0.6716 - val_accuracy: 0.8016
Epoch 8/200
125/125 [==============================] - 1s 11ms/step - loss: 0.5353 - accuracy: 0.8421 - val_loss: 0.6505 - val_accuracy: 0.8067
Epoch 9/200
125/125 [==============================] - 1s 11ms/step - loss: 0.5160 - accuracy: 0.8477 - val_loss: 0.6219 - val_accuracy: 0.8170
Epoch 10/200
125/125 [==============================] - 1s 11ms/step - loss: 0.4984 - accuracy: 0.8526 - val_loss: 0.6109 - val_accuracy: 0.8198
```

On successful execution of above command , our project is trained.

Implementation and System Testing

After all phase have been perfectly done, the system will be executed in terminal and the system can be used.

System Testing:

The goal of the system testing process was to determine all faults in our project .The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing
2 .Integration testing

Unit Testing

Unit testing is commenced when a unit has been created and effectively reviewed .In order to test a single module we need to provide a complete environment i.e. besides the section we would require The procedures belonging to other units that the unit under test calls Non local data structures that module accesses .A procedure to call the functions of the unit under test with appropriate parameters Here we test each and every line of code.

Integration Testing

In the Integration testing we test various combination of the project module by providing the input.

The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

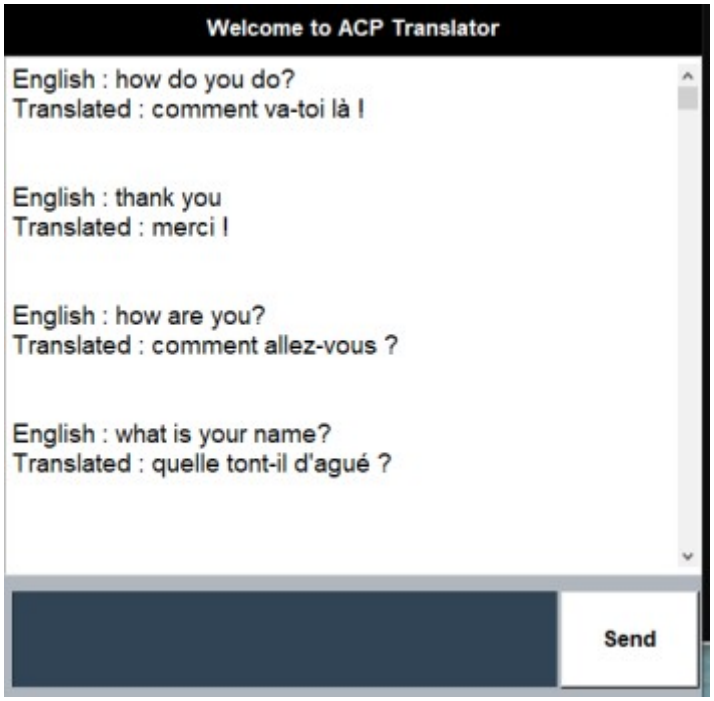
EVALUATION

Exection: By runnig the following command.

python LangTransGui.py

Our project will execute.

OUTPUT:



CONCLUSION

ML methods are popularly used for language translation now a days. In this project English to French Language Translation is done by using ML Techniques.Moreover not only we use technology, but it also fully offline system which is user friendly and reliable.

REFERENCES:

https://www.codegrepper.com

https://numpy.org/doc/stable/

https://scikit-learn.org