NIRF-2024
Engineering Rank Band (151-200)
Pharmacy Rank - 77
Innovation Rank Band (11-50)

www.kiet.edu
Delhi-NCR, Ghaziabad

KIET GROUP OF INSTITUTIONS
Connecting Life with Learning

Accredited by
NAAC with Grade "A+"

Department of Computer Application

LANGUAGE TRANSLATION MODEL

A PROJECT REPORT
FOR
INTRODUCTION TO AI(AI101B)
SESSION(2024-25)

SUBMITTED BY
SHANTANU YADAV(202410116100193)
SHIVAM CHATURVEDI(202410116100198)
SONAM DOBRIYAL(202410116100210)
VINEET KUMAR(202410116100246)

Submitted to
KOMAL SALGOTRA
ASS. PROFESSOR

INTRODUCTION

In today's interconnected world, language should no longer be a barrier to communication and collaboration. As part of a project assigned by my teacher—where each student was given a different unique idea to work on—I was given the opportunity to develop a Language Translation AI. Although the idea was provided, I had full freedom to research, design, and implement the solution in my own way, applying my creativity and technical skills throughout the development process.

The goal of this project was to build an AI-based tool that can accurately translate text from one language to another while maintaining the context and natural flow of communication. Unlike simple word-to-word translation systems, this project uses Natural Language Processing (NLP) and machine learning to deliver context-aware and grammatically correct translations across multiple languages.

From understanding how language models work to preprocessing data, training the model, and developing the user interface, I took complete responsibility for the execution of the project. I used modern tools and frameworks to create a reliable and user-friendly platform that can help individuals, students, travelers, and professionals communicate without language restrictions.

This project not only helped me enhance my knowledge in artificial intelligence and web development but also taught me how to bring an idea to life through independent research, problem-solving, and continuous improvement. It stands as an example of how a guided idea can turn into a powerful real-world solution with the right approach and dedication.

Department of Computer Application

METHODOLOGY

1. **Data Collection**: A dataset containing English–French sentence pairs was used. The source is the ManyThings.org version of the Tatoeba dataset.

2. **Data Preprocessing**:
    1. Lowercasing, trimming, and adding special start/end tokens.
    2. Tokenization and padding of input and output sequences.

3. **Model Architecture**:
    1. Encoder: Embedding layer + LSTM
    2. Decoder: Embedding + LSTM + Dense layer with Softmax

4. **Training**:
    1. Teacher forcing technique
    2. Loss function: Categorical crossentropy
    3. Optimizer: RMSProp

5. **Inference**:
    1. Encoder encodes input sentence into states.
    2. Decoder predicts one word at a time until token.

CODE

```
import string
import re
import numpy as np
from numpy import array, argmax, random, take
import pandas as pd
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding, RepeatVector
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import load_model
from tensorflow.keras import optimizers
```

```
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_colwidth',200)
```

```
data_path='fra.txt'
with open(data_path, 'r', encoding='utf-8') as f:
  lines = f.read()

lines
```

```
def to_lines(text):
  sents=text.strip().split('\n')
  sents=[i.split('\t') for i in sents ]
  return sents
```

```
fra_eng=to_lines(lines)
fra_eng[:5]
```

```
[['Go.',
  'Va !',
  'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev)'],
 ['Go.',
  'Marche.',
  'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8090732 (Micsmithel)'],
 ['Go.',
  'En route !',
  'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8267435 (felix63)'],
 ['Go.',
  'Bouge !',
  'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #9022935 (Micsmithel)'],
 ['Hi.',
  'Salut !',
  'CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji)']]
```

```
fra_eng=array(fra_eng)
fra_eng[:5]
```

```
array([['Go.', 'Va !',
        'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev)'],
       ['Go.', 'Marche.',
        'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8090732 (Micsmithel)'],
       ['Go.', 'En route !',
        'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8267435 (felix63)'],
       ['Go.', 'Bouge !',
        'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #9022935 (Micsmithel)'],
       ['Hi.', 'Salut !',
        'CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji)']],
      dtype='<U101')
```

```
fra_eng.shape
```

```
(50875, 3)
```

```
fra_eng=fra_eng[:100000,:]
fra_eng=fra_eng[:,[0,1]]
fra_eng[:5]
```

```
array([['Go.', 'Va !'],
       ['Go.', 'Marche.'],
       ['Go.', 'En route !'],
       ['Go.', 'Bouge !'],
       ['Hi.', 'Salut !']], dtype='<U101')
```

```
fra_eng[:,0]=[s.translate(str.maketrans('','',string.punctuation)) for s in fra_eng[:,0]]
fra_eng[:,1]=[s.translate(str.maketrans('','',string.punctuation)) for s in fra_eng[:,1]]
fra_eng[:5]
```

```
array([['Go', 'Va '],
       ['Go', 'Marche'],
       ['Go', 'En route '],
       ['Go', 'Bouge '],
       ['Hi', 'Salut ']], dtype='<U101')
```

```
for i in range(len(fra_eng)):
  fra_eng[i,0]=fra_eng[i,0].lower()
  fra_eng[i,1]=fra_eng[i,1].lower()

fra_eng
```

```
array([['go', 'va '],
       ['go', 'marche'],
       ['go', 'en route '],
       ...,
       ['its not a classroom', 'ce nest pas une salle de classe'],
       ['its not a good time', 'ça tombe assez mal'],
       ['its not about money', 'il ne sagit pas dargent']], dtype='<U101')
```

```
def tokenization(lines):
  tokenizer=Tokenizer()
  tokenizer.fit_on_texts(lines)
  return tokenizer

eng_tokenizer=tokenization(fra_eng[:,0])
eng_vocab_size=len(eng_tokenizer.word_index)+1

eng_length=8
print('English vocabulary size:' , eng_vocab_size)
```

```
English vocabulary size: 6028
```

```
fra_tokenizer=tokenization(fra_eng[:,1])
fra_vocab_size=len(fra_tokenizer.word_index)+1
fra_length=8
print('French Vocabulary Size: ', fra_vocab_size)
```

```
French Vocabulary Size:  13842
```

```
def encode_sequences(tokenizer,length, lines):
  seq=tokenizer.texts_to_sequences(lines)
  seq=pad_sequences(seq,maxlen=length,padding='post')
  return seq
```

```
from sklearn.model_selection import train_test_split
train, test= train_test_split(fra_eng,test_size=0.2,random_state=12)
trainX=encode_sequences(fra_tokenizer,fra_length,train[:,1])
trainY= encode_sequences(eng_tokenizer,eng_length,train[:,0])

testX=encode_sequences(fra_tokenizer,fra_length,test[:,1])
testY=encode_sequences(eng_tokenizer, eng_length, test[:,0])
```

```
def define_model(in_vocab,out_vocab, in_timesteps,out_timesteps, units):
  model=Sequential()
  model.add(Embedding(in_vocab, units, input_length=in_timesteps, mask_zero=True))
  model.add(LSTM(units))
  model.add(RepeatVector(out_timesteps))
  model.add(LSTM(units,return_sequences=True))
  model.add(Dense(out_vocab, activation='softmax'))
  return model
```

```
model=define_model(fra_vocab_size, eng_vocab_size, fra_length, eng_length, 512)
rms=optimizers.RMSprop(learning_rate=0.001)
model.compile(optimizer=rms,loss='sparse_categorical_crossentropy')
```

```
history=model.fit(trainX, trainY.reshape(trainY.shape[0], trainY.shape[1],1),
                  epochs=30, batch_size=512, validation_split=0.2)
```

```
Epoch 1/30
64/64 ━━━━━━━━━━━━━━━━━━━━ 963s 15s/step - loss: 5.6927 - val_loss: 3.0776
Epoch 2/30
64/64 ━━━━━━━━━━━━━━━━━━━━ 948s 14s/step - loss: 2.9570 - val_loss: 2.7746
Epoch 3/30
26/64 ━━━━━━━━━━━━━━━━━━━━ 8:35 14s/step - loss: 2.7630
```

```
preds_probs = model.predict(testX)
preds = np.argmax(preds_probs, axis=-1)
```

```
preds
```

```
array([[17,  2, 11, ...,  0,  0,  0],
       [ 1, 15,  4, ...,  0,  0,  0],
       [14,  5,  4, ...,  0,  0,  0],
       ...,
       [14, 15,  4, ...,  0,  0,  0],
       [39, 89,  9, ...,  0,  0,  0],
       [13,  2,  2, ...,  0,  0,  0]])
```

```
def get_word(n,tokenizer):
  for word,index in tokenizer.word_index.items():
    if index==n:
      return word
  return None
```

```
preds_text=[]
for i in preds:
  temp=[]
  for j in range(len(i)):
    t=get_word(i[j],eng_tokenizer)
    if j>0:
      if(t==get_word(i[j-1], eng_tokenizer)) or (t==None):
        temp.append('')
      else:
        temp.append(t)
    else:
      if(t==None):
        temp.append('')
      else:
        temp.append(t)
  preds_text.append(' '.join(temp))
```

```
pred_df=pd.DataFrame({'actual': test[:,0], 'predicted': preds_text})
```

```
pred_df.sample(10)
```

```
pred_df.sample(10)
```

	actual	predicted
4522	are you courageous	are you
8109	that was a secret	he was a
5487	tom is in luck	tom is a
9812	dont be naive	were
3857	theyre very happy	theyre are
6536	everyone watched	youre not
2536	are we sinking	you
2001	whats it good for	we me
642	are you ambitious	are you
3001	i remain doubtful	i am

```
import matplotlib.pyplot as plt


plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```