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+"

# INTRODUCTION TO AI(AI101B)

## Even Semester

## Session 2024-25

### LANGUAGE TRANSLATION WITH SEQUENCE MODEL

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

**Project Supervisor:**

Ms. Komal Salgotra
Assist. Professor

# Content

- Introduction
- Objective
- Methodology
- Code
- Outcomes
- References

# Introduction

Language is the foundation of communication, and the ability to automatically translate one language to another is a significant milestone in artificial intelligence and natural language processing. This project explores the use of sequence models, particularly sequence-to-sequence (Seq2Seq) models with LSTM layers, to perform language translation. By training on English–French sentence pairs, the model learns to map input sequences in French to target sequences in English. The project demonstrates how neural networks can be used to learn patterns in linguistic data and apply them to real-world translation tasks.

# Objective of the Project

The objective of this project is to build a machine learning-based model that translates English sentences into French using a sequence-to-sequence architecture. The model utilizes LSTM layers to capture the temporal dependencies in sentence structures and generates meaningful translations. This project aims to train the model on real French-English translation datasets.

# 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**:

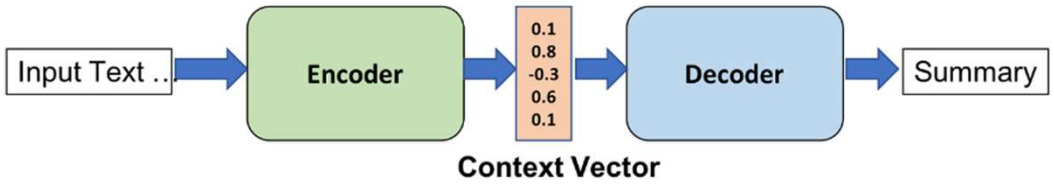
1. Lowercasing, trimming, and adding special start/end tokens.
2. Tokenization and padding of input and output sequences.

**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

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

```
[2]  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
```

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

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

     lines
```

```
'Go.\tVa !\tCC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev)\nGo.\tMarc
ba.org #2877272 (CM) & #8090732 (Micsmithel)\nGo.\tEn route !\tCC-BY 2.0 (France) Attribution: tatoeba
o.\tBouge !\tCC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #9022935 (Micsmithel)\nHi.\tS
```

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

```
[7]  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)']]
```

```
[8]  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)'],
```

```
fra_eng.shape
```

```
(232736, 3)
```

```
fra_eng=fra_eng[:50000,:]
```

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

```
array([['Go.', 'Va !'],
       ['Go.', 'Marche.'],
       ['Go.', 'En route !'],
       ['Go.', 'Bouge !'],
       ['Hi.', 'Salut !']], dtype='<U349')
```

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

```
array([['Go', 'Va '],
       ['Go', 'Marche'],
       ['Go', 'En route '],
       ['Go', 'Bouge '],
       ['Hi', 'Salut ']], dtype='<U349')
```

```
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()
```

```
[14] 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: 1907
```

```
[15] 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:  4422
```

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

```
[17] 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')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecate
  warnings.warn(
```

```
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
63/63 ━━━━━━━━━━━━━━━━━━━━ 20s 219ms/step - loss: 5.6975 - val_loss: 3.1559
Epoch 2/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 212ms/step - loss: 2.9637 - val_loss: 2.7826
```

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

```
58/58 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
```

```
[23] preds
```

```
array([[ 1,  5,  0, ...,  0,  0,  0],
       [ 1,  7,  0, ...,  0,  0,  0],
       [ 2,  2,  0, ...,  0,  0,  0],
       ...,
       [ 2,  2,  0, ...,  0,  0,  0],
       [ 1,  6,  0, ...,  0,  0,  0],
       [ 1, 32,  0, ...,  0,  0,  0]])
```

```
[24] 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))
```

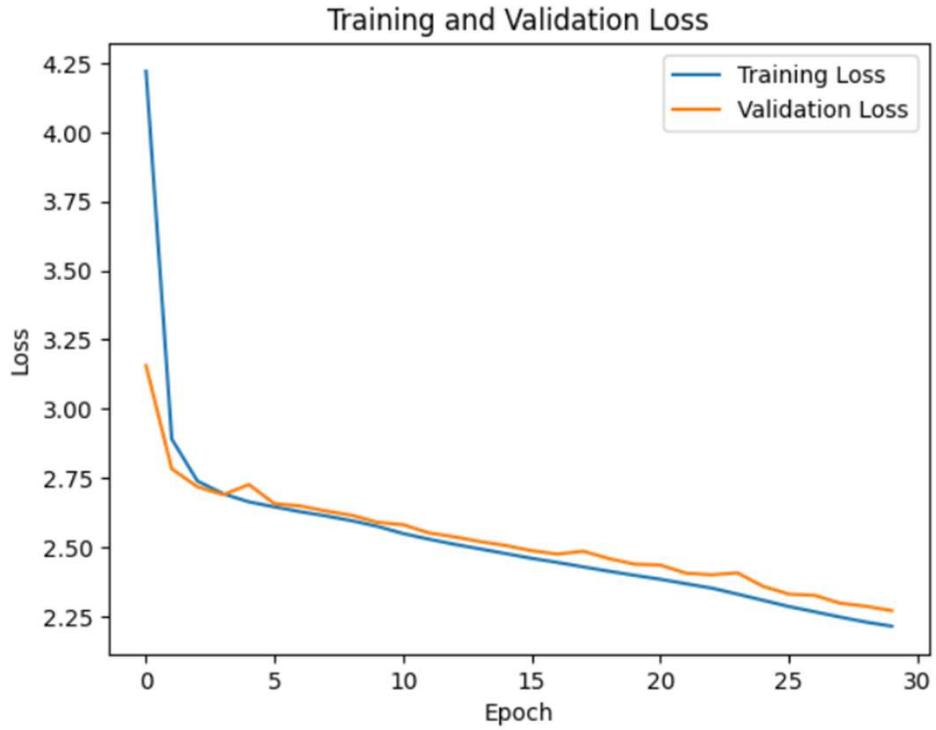
```
pred_df=pd.DataFrame({'actual': test[:,0], 'predicted': preds_text})
```

## Output

```
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()
```



# Outcome

This project successfully demonstrates how sequence models can be used for basic language translation. The model trained on sample data can generate understandable French translations for English sentences. The project lays the foundation for scaling up to more advanced NLP tasks like multilingual translation, attention mechanisms, or Transformer-based models.

# References

1. ManyThings.org English–French Dataset: http://www.manythings.org/
2. TensorFlow Documentation – https://www.tensorflow.org/
3. Keras Seq2Seq Guide – https://keras.io/examples/nlp/lstm_seq2seq/