A MINI PROJECT REPORT

ON

LANGUAGE TRANSLATOR

Submitted by

KAAVIYASREE.SK

In partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KARPAGAM COLLEGE OF ENGINEERING

NOVEMBER 2022

BONAFIDE CERTIFICATE

Roll No. 20I125

Certified that this mini project report **LANGUAGE TRANSLATOR** is the bonafide work of Ms.**KAAVIYASREE SK** of V semester B.Tech.Artificial Intelligence and Data Science during the academic year 2022 in the **18ID24/Deep Learning Methods Using Python.**

Faculty in-charge **Head of the Department**

ACKNOWLEDGEMENT

We express our sincere thanks to Karpagam educational and charitable trust for providing necessary facilities to bring out the project successfully. We felt great to record our thanks to the chairman Dr.R.VASANTHAKUMAR, B.E.,(Hons),D.Sc.for all his support and ray of strengthening hope extended.

It is a moment of immense pride for us to reveal our profound thanks to our respected principal, Dr.P.VIJAYAKUMAR, M.EPh.D. who happens to be a striving force in all our endeavors.

We express our sincere thanks to our Mr. Manikandan Sundaram M. Tech, (Ph. D) Head of the Department of Artificial Intelligence and Data Science for providing an opportunity to work on this project. His valuable suggestions helped us a lot to do this project.

A word of thanks would not be sufficient for the work of our project guide Ms. Sathya.K M.E Department of Artificial Intelligence and Data Science whose efforts and inspiration lead us through every trying circumstance.

We deeply express our gratitude to all the members of the faculty of the department of Artificial Intelligence and Data Science for the encouragement, which we received throughout the semester.

VISION

To become an integrative department through creativity and innovation in developing systems with Artificial Intelligence and Data Science to serve society.

MISSION

- Engage collaborative activities with the industry to produce industry-ready and futureready talent.
- Foster the spirit of lifelong learning in students through practical and social exposure beyond the classroom.
- Provide multi-disciplinary research and innovation driven academic environment.
- Practice and instill the highest standards of professional ethics, transparency and accountability at all levels.

Programme Educational Objectives (PEOs)

- **PEO1:** Graduates will be able to demonstrate technical skills, competency in Artificial Intelligence and Data Science, exhibit proficiency and team management capability with proper communication in the job environment.
- **PEO2:** Graduates will be able to work effectively in multidisciplinary engineering projects and support the growth of the economy of a nation by becoming entrepreneurs with a lifelong learning practice.
- **PEO3:** Graduates will be able to carry out the research in the contemporary areas of Artificial Intelligence, Data Science and address the basic needs of the society.

Program Outcomes

- **PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend

  and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12: Life-long learning** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Demonstrate the knowledge of Artificial Intelligence, Machine Learning and Data Science for designing and developing intelligent systems.
- **PSO2:** Implement Artificial Intelligence and data science methods for solving a problem in a multi-disciplinary area and design novel algorithms for a successful career.

INDEX

ABSTRACT

In today's world there are many people who are facing the problem of language translator.In this project, I build a deep neural network that functions as part of a machine translation pipeline. The pipeline accepts French text as input and returns the English translation. The goal is to achieve the highest translation accuracy possible.Language translation is a method of converting the source sentence from one natural language to another natural language using computerized systems and human assistance is not required. In this report, I have introduced you to a Deep learning project on language translation with Python programming language.Deep Learning is a recently used approach for language translation. Unlike traditional machine translation, neural machine translation is a better choice for more accurate translation and also offers better performance. DNN can be used to improve traditional systems to make them more efficient.Different deep learning techniques and libraries are needed to develop a better language translation system. RNN, LSTM, etc. are used to train the system which will convert the sentence from the source language to the target language.Train a sequence to sequence model on a dataset of English and French sentences that can translate new sentences from English to French.Adapting the appropriate networks and deep learning strategies is a good choice, as it has turned the system to maximize the accuracy of the translation system relative to others.

Keywords: LSTM, Encoder, Decoder, Language translator

INTRODUCTION

Language Translation (LT) is an empirical approach to the known process of Machine Translation that uses a large artificially constructed neural network to predict or to know the occurrence of a large sequence of words, basically modeling or generating the entire sentences in a single integrated model. It has gained adoption in many large-scale functionalities. LT systems take advantage of continuous representations that greatly ease the sparsity problem, and make use of much larger contexts, thus lessening the locality problem. Many issues and shortcomings of the traditional machine translation systems is eradicated by the new approach i.e. LT. Deep Neural Machine Translation i.e. Deep LT is an extension of NMT. Both of them use a large neural network with the only difference that exists is that deep neural machine translation processes multiple neural network layers instead of just one as it incorporates higher proficiency compared to LT. The word sequence modeling was at first typically done using a RNN Recurrent Neural Network. A bidirectional recurrent neural network which was used as an encoder used by the neural network to encode a source sentence and a second RNN known as a decoder, that is used to predict words or sentences in the target language was used.

Requirement Specifications

HARDWARE REQUIREMENTS

RAM : 128 MB

Hard Disk : 20GB

Monitor : 15” Colour monitor

Key Board : 108 Keys

SOFTWARE REQUIREMENTS

Operating System : Windows 10

Language : Python

Software : Google Colab Notebook

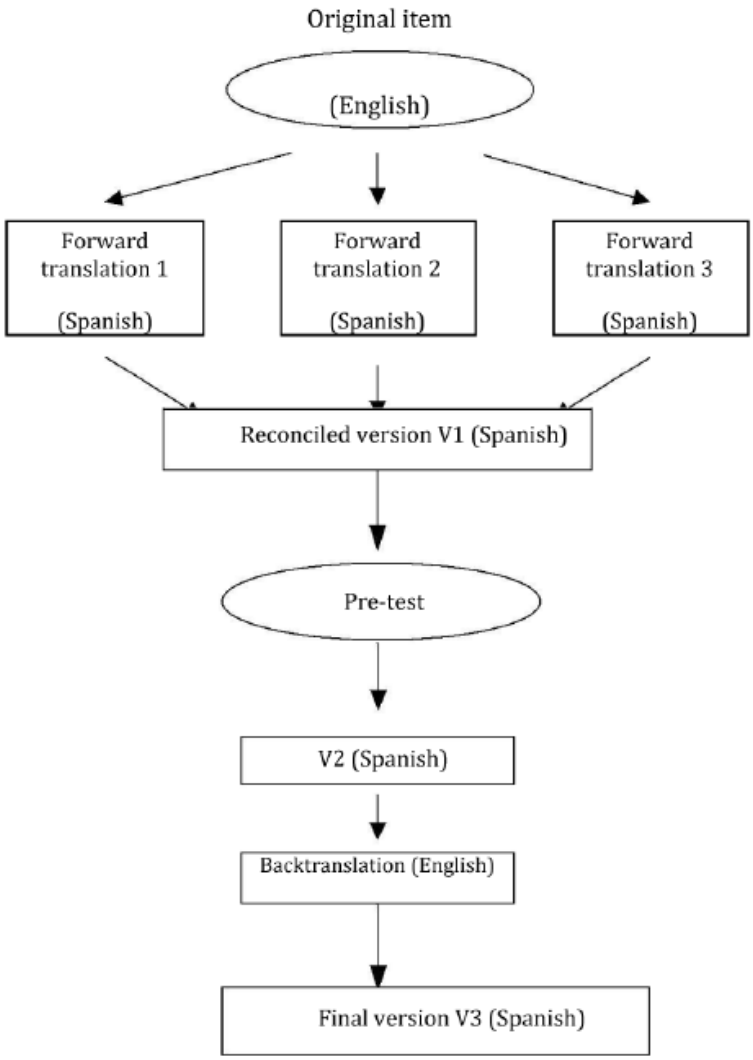
Make sure you have installed all the following necessary libraries:

- Tensorflow
- Keras
- Numpy
- Torch

DATASET DESCRIPTION:

As we have used the French text dataset which contains two columns , where the first column represents the English language words and the second column represents the French language words which is going to be translated with the help of the concepts Deep learning and Neural Network.

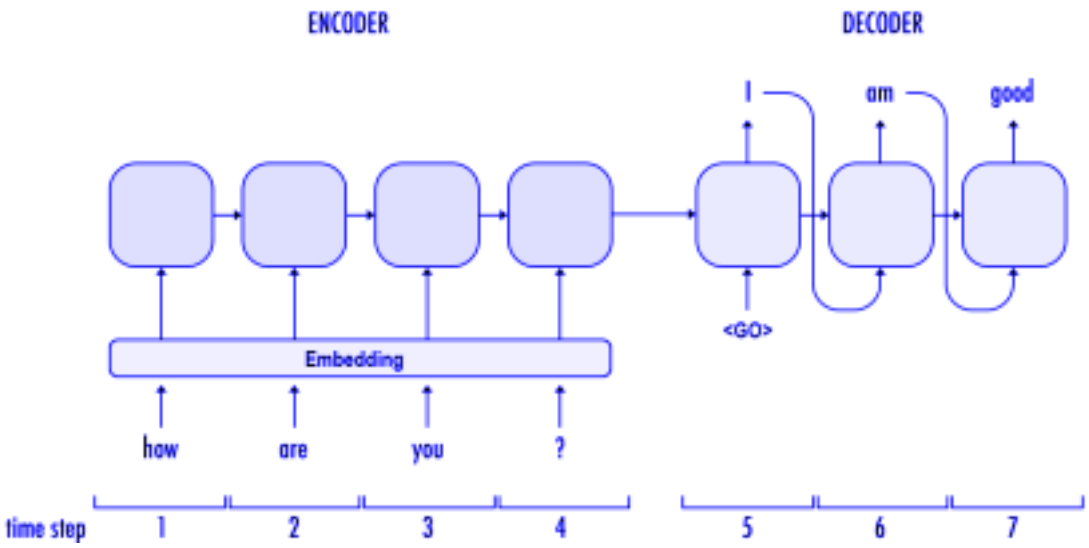
FLOWCHART:



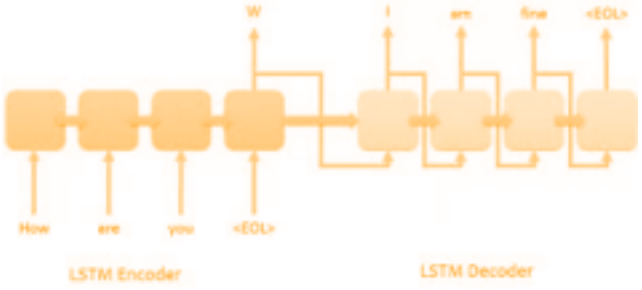
METHODOLOGY:

For this research work Language Translation (LT) is taken as methodology. LT introduced a new approach with the capability of addressing many shortcomings of traditional machine translation systems. The Benefit or advantage of LT lies in its ability to learn precisely, i.e. mapping of input text to associated output text. Its architecture consists of two recurrent neural networks (RNNs), one is

used to absorb the input text sequence i.e. encoder and one is used to generate translated output text i.e. decoder.



An encoder converts an input sentence into a "meaning" vector which is passed through a decoder to provide a translation.



Specifically, an LT system first reads the input sentence using an encoder to build a “thought” vector , a sequence or series of numbers that represents the sentence meaning; a decoder, then, processes and

provides the sentence vector to produce a translation, often referred to as the encoder-decoder architecture. LT models vary in terms of their exact architectures. An obvious choice for the sequential data is the RNN i.e. Recurrent Neural Network, which is used by most LT models. Usually an RNN is used for both the encoder and decoder. These models, however, differ in terms of: (a) **directionality** –i.e. can be unidirectional or bidirectional; (b) **depth** – i.e. can be single- or multi-layer; (c) **type** – often either a vanilla RNN, a Long Short-term Memory (LSTM), or a gated recurrent unit (GRU)

SOURCE CODE:

```
import tensorflow as tf
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount('/content/drive')
lines = pd.read_table('/content/drive/My Drive/Colab Notebooks/french.txt', names=['English',
'French'])
lines = lines[:8000]
lines.sample(5)
lines.shape
lines.English = lines.English.apply(lambda x: x.lower())
lines.French = lines.French.apply(lambda x: x.lower())
import re
lines.English = lines.English.apply(lambda x: re.sub("""",
    '', x)).apply(lambda x: re.sub(",", 'COMMA', x))
lines.French = lines.French.apply(lambda x: re.sub("""",
    '', x)).apply(lambda x: re.sub(",", 'COMMA', x))
```

```
    import string
    exclude = set(string.punctuation)
    lines.English = lines.English.apply(lambda x: ''.join(ch for ch in x if ch not in exclude))
    lines.French = lines.French.apply(lambda x: ''.join(ch for ch in x if ch not in exclude))
from string import digits
remove_digits = str.maketrans('', '', digits)
lines.English = lines.English.apply(lambda x: x.translate(remove_digits))
lines.French = lines.French.apply(lambda x: x.translate(remove_digits))
lines.sample(7)
# applying start and end tokens in french sentences
lines.French = lines.French.apply(lambda x: 'START_' + ' ' + x + ' ' + '_END')
lines.head()
# collecting all unique english words to create a vocabulary
all_English_words = set()
for eng in lines.English:
  for word in eng.split():
    if word not in all_English_words:
      all_English_words.add(word)

# collecting all unique french words to create a vocabulary
all_French_words = set()
for fre in lines.French:
  for word in fre.split():
    if word not in all_French_words:
      all_French_words.add(word)
# printing length of words in each language
print('length of English words: ', len(all_English_words))
print('length of French words: ', len(all_French_words))
# getting maximum sentence length of english sentences
length_list = []
for l in lines.English:
```

```
  length_list.append(len(l.split(' ')))

max_input_length = np.max(length_list)
print('max_input_length: ', max_input_length)
    # getting maximum sentence length of french sentences
    length_list = []
    for l in lines.French:
      length_list.append(len(l.split(' ')))

    max_output_length = np.max(length_list)
    print('max_output_length: ', max_output_length)
       # making a list of all input and output words and sorting them out
       input_words = sorted(list(all_English_words))
       output_words = sorted(list(all_French_words))
       print('all input words: ', input_words)
       print('all output words: ', output_words)

       #getting total tokens(words) from input and output
       num_encoder_tokens = len(all_English_words)
       num_decoder_tokens = len(all_French_words)
       print('encoder tokens: ', num_encoder_tokens)
       print('decoder tokens: ', num_encoder_tokens)
# getting index for words as these indexes will behave as words for machine interactions
input_token_index = dict([(word,i) for i,word in enumerate(input_words)])
output_token_index = dict([(word,i) for i,word in enumerate(output_words)])

print('input token index: ', input_token_index)
print('output token index: ', output_token_index)
# creating arrays of input and output data
encoder_input_data = np.zeros((len(lines.English), max_input_length), dtype='float32')
decoder_input_data = np.zeros((len(lines.French), max_output_length), dtype='float32')
```

```
#one hot encoding the target data as Dense layer only gives one output through softmax layer
decoder_target_data = np.zeros((len(lines.French), max_output_length, num_decoder_tokens))
print(encoder_input_data.shape)
print(decoder_input_data.shape)
print(decoder_target_data.shape)
# putting all the integer values in input, output data and target data
for i,(input_text, output_text) in enumerate(zip(lines.English, lines.French)):
  for t, word in enumerate(input_text.split()):
    encoder_input_data[i,t] = input_token_index[word]
  for t,word in enumerate(output_text.split()):
    decoder_input_data[i,t] = output_token_index[word]
    # as decoder target data is ahead of decoder input data, it will not include start_
character(which will be given to decoder model at prediction)
    if t > 0:
      decoder_target_data[i,t-1,output_token_index[word]] = 1
     print("encoder input data: ", encoder_input_data[1])
     print('decoder input data: ', decoder_input_data[1])
     print('decoder target data: ',decoder_target_data[1])
     print('shape of sample decoder target data: ', decoder_target_data[1].shape)
         from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
         from tensorflow.keras.models import Model
         from tensorflow.keras.utils import plot_model
             # setting hyperparameters
             embedding_size = 120
             lstm_dim = 324
         # building model for training stage
         #encoder model

         encoder_inputs = Input(shape=(None,))
         en_x = Embedding(num_encoder_tokens, embedding_size)(encoder_inputs)
```

```
        encoder = LSTM(lstm_dim, return_state=True)
        encoder_outputs, state_h, state_c = encoder(en_x)
        encoder_states = [state_h, state_c]
    # decoder model

    decoder_inputs = Input(shape=(None,))
    final_dex = Embedding(num_decoder_tokens, embedding_size)(decoder_inputs)

    decoder_lstm = LSTM(lstm_dim, return_sequences=True, return_state=True)

    decoder_outputs, _, _ = decoder_lstm(final_dex, initial_state=encoder_states)

    decoder_dense = Dense(num_decoder_tokens, activation='softmax')

    decoder_outputs = decoder_dense(decoder_outputs)
    model = Model([encoder_inputs, decoder_inputs],
    decoder_outputs)
model.compile(optimizer='rmsprop',
            loss = 'categorical_crossentropy',
            metrics=['accuracy'])

model.summary()
r = model.fit([encoder_input_data, decoder_input_data], decoder_target_data, batch_size=64,
epochs=30, validation_split=0.10)
#Inference Stage

#encoder model
encoder_model = Model(encoder_inputs, encoder_states)
encoder_model.summary()
#decoder model
decoder_state_input_h = Input(shape=(lstm_dim,))
```

```
decoder_state_input_c = Input(shape=(lstm_dim,))
decoder_state_inputs = [decoder_state_input_h, decoder_state_input_c]

final_dex2 = Embedding(num_decoder_tokens, embedding_size)(decoder_inputs)

decoder_outputs2, state_h2, state_c2 = decoder_lstm(final_dex2, initial_state=decoder_state_inputs)
decoder_states2 = [state_h2, state_c2]
decoder_outputs2 = decoder_dense(decoder_outputs2)

decoder_model = Model([decoder_inputs] + decoder_state_inputs, [decoder_outputs2] +
decoder_states2)
# reversing the word index dictionary to get words from index values
reverse_input_char_index = dict((i,char) for char, i in input_token_index.items())
reverse_output_char_index = dict((i,char) for char, i in output_token_index.items())
print(reverse_input_char_index)
print(reverse_output_char_index)
# function to predict translation
def decode_seq(input_seq):
  state_values = encoder_model.predict(input_seq)

  target_seq = np.zeros((1,1))

  target_seq[0,0] = output_token_index['START_']

  stop_condition = False
  decoded_sentence = ''

  while not stop_condition:
    output_tokens, h, c = decoder_model.predict([target_seq] + state_values)

    sampled_token_index = np.argmax(output_tokens[0,-1,:])
```

```
    sampled_char = reverse_output_char_index[sampled_token_index]

    decoded_sentence += ' ' + sampled_char

    if(sampled_char == '_END' or len(decoded_sentence) > 52):
      stop_condition = True

    target_seq = np.zeros((1,1))
    target_seq[0,0] = sampled_token_index

    state_values = [h,c]

  return decoded_sentence
import tensorflow as tf
import numpy as np

tf.random.set_seed(42)

# Create some regression data
X_regression = np.expand_dims(np.arange(0, 1000, 5), axis=1)
y_regression = np.expand_dims(np.arange(100, 1100, 5), axis=1)

# Split it into training and test sets
X_reg_train = X_regression[:150, :]
X_reg_test = X_regression[150:, :]

y_reg_train = y_regression[:150, :]
y_reg_test = y_regression[150:, :]

tf.random.set_seed(42)
```

```
# Recreate the model
model_3 = tf.keras.Sequential([
  tf.keras.layers.Dense(100),
  tf.keras.layers.Dense(10),
  tf.keras.layers.Dense(1)
])

# Change the loss and metrics of our compiled model
model_3.compile(loss=tf.keras.losses.mae, # change the loss function to be regression-specific
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
                metrics=['mae']) # change the metric to be regression-specific

# Fit the recompiled model
model_3.fit(X_reg_train, y_reg_train, epochs=100)

model_3.predict(X_reg_test)
!pip install keras
# testing the model for a sample from existing data
for seq_index in [224, 354, 181, 17, 256, 310]:
  input_seq = encoder_input_data[seq_index:seq_index+1]
  decoded_sentence = decode_seq(input_seq)
  print('----')
  print('Input_sentence: ', lines.English[seq_index:seq_index+1])
  print('decoded sentence: ', decoded_sentence)
```

OUTPUT :

	English	French
387	they treat their employees well	ils traitent bien leurs employés
319	there wasnt anyone in the room	personne ne se trouvait dans la pièce
199	since it will be cold soonCOMMA it might be ni...	puisquil fera bientôt froidCOMMA ça serait cho...
138	i realize i may not be the most desirable man ...	je me rends compte que je ne suis peutêtre pas...
334	theres nowhere for you to hide	il ny a nulle part où tu puisses te cacher
385	they are from the united states	elles sont des étatsunis
131	i cant believe that you arent at least willing...	je narrive pas à croire que vous ne soyez pas ...

```
all input words:  ['a', 'abandoned', 'able', 'abolished', 'about', 'aboutCOMMA', 'above', 'abroad', 'acceler
all output words:  ['START_', '_END', 'a', 'abandonnèrent', 'abandonné', 'abandonnée', 'abattu', 'aboli', 'a
encoder tokens:  1609
decoder tokens:  1609
```

```
encoder input data:  [1597. 1553. 1118.    0.  297.  442.   35. 1602.   14.  629.  141.   21.
  199.   70.   26.    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.]
decoder input data:  [0.000e+00 1.907e+03 4.400e+01 1.476e+03 1.830e+03 3.320e+02 1.214e+03
 3.840e+02 8.200e+01 8.960e+02 4.330e+02 1.903e+03 3.250e+02 1.214e+03
 1.828e+03 2.200e+01 1.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00]
decoder target data:  [[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.]]
shape of sample decoder target data:  (56, 1985)
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
----
Input_sentence:  354    these peaches arent very sweet
Name: English, dtype: object
decoded sentence:   il y a de de de de de _END
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
----
Input_sentence:  181    its very easy to sound natural in
Name: English, dtype: object
decoded sentence:   il y a de de de _END
```

RESULT AND DISCUSSION:

As we have done an implementation of translating a word from English to Russian with the help of some specific libraries and used a LSTM layer with the basic concepts of deep learning , Neural network and a part of Recurrent Neural Network. When the input sentence is given it gets converted into Russian word by the trained model which gets fit and evaluated and gives the best accuracy with the least loss .

CONCLUSION:

In practice, however, LT systems used to be better in accuracy than phrase-based translation systems, especially when training on very large-scale datasets as used for the very best publicly available translation systems. Weaknesses of LT are responsible for this gap:

slower training and inference speed, ineffectiveness in dealing with rare words, and sometimes failure to translate all words in the source sentence. Finally, it takes a considerable amount of time and computational resources to train an LT system on a large scale translation dataset, thus slowing the rate of experimental turnaround time and innovation. For inference, they are generally much slower than phrase-based systems due to the large number of parameters used.

REFERENCES:

[1] A. Waibel, A. Badran, A. W Black, R. Frederking, D. Gates, A. Lavie, K. Lenzo, L. Tomokiyo, J. Reichert, T. Schultz, D. Wallace, M. Woszczyna and J. Zhang, “Speechalator: two-way speech-to-speech translation on a consumer PDA”, EUROSPEECH 2003 – GENEVA.

[2] U. Kheradia and A. Kondwilkar, ”Speech To Speech Language Translator”, International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012 1 ISSN 2250-3153.

[3] D. Bahdanau, K. Cho and Y. Bengio, “NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE”, Published as a conference paper at ICLR 2015

[4] L. Jain and P. Agrawal, “English to Sanskrit Transliteration: an effective approach to design Natural Language Translation Tool”, International Journal of Advanced Research in Computer Science, Volume 8, No. 1, Jan-Feb 2017

[5] P. Koehn and R. Knowles, “Six Challenges for Neural Machine Translation”, Proceedings of the First Workshop on Neural Machine Translation, pages 28–39(2017).

[6] K. Revanuru, K. Turlapaty, S. Rao, “Neural Machine Translation

of Indian Languages" Published as a Conference paper at COMPUTE 2017: 10th Annual ACM India Conference, At Bhopal, India, November 2017

[7] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen and N. Thorat, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation",(2017)