Translatish - Video Translation System

A Project Report

Submitted By:

Aanshi Patwari(AU1841004)

Dipika Pawar(AU1841052)

Mayankkumar Tank(AU1841057)

Rahul Chocha(AU1841076)

Yash Patel(AU1841125)

at

School of Computer Studies(SCS)

Ahmedabad, Gujarat

Table of Contents

1. Introduction

1.1. Problem Statement

- Conversion of videos in English language into our national language(Hindi language) such that it can reach a wider audience of the country.

1.2. Project Overview/Specifications

- The project proposes the use of neural machine translation techniques to convert the videos in English language into Hindi language.
- The technique involves the steps as follows:
    - Input : Video in English language
    - Process :
        - Fetch audio from video
        - Convert audio to text
        - Translate english text to hindi text
    - Output: Hindi text

1.3. Motivation

- India is a multilingual country where more than 1600 languages are spoken belonging to more than 4 different language families; with Hindi being the national language.
- This beauty of language transcends the boundaries of the different states and cultures in our country. Learning a language other than the mother language is a huge advantage. But a path to multilingualism is a never ending process for most of the people of our country.
- Majority of the population of our country finds it difficult to communicate in English and understand English language videos. So, translation of such videos can become useful.
- Furthermore, by converting visual content(i.e. videos) into understandable language(here, hindi) becomes an important step for understanding the relation between visual and linguistic information which are the richest interaction modalities available to humans.

2. Literature Review

2.1. Existing System

- Rule Based Machine Translation(RBMT)
    - These models are based on the idea of use of available dictionaries and rules of a particular language for translating into another language.
    - These models are prepared manually leading to need of large human task force who can understand both the languages(i.e. linguists)
- Example Based Machine Translation(EBMT)
    - These models are based on the idea of using the available set of translations to continue the translation from one language to another.
    - These models have the limitations of having a limited number of examples being available.
- Static Machine Translation(SMT)
    - Word based
        - Comparing similar words in both languages and understanding the pattern by doing the same operation a number of times using the concept of bag-of-words.
        - This model had a limitation of not being able to consider the order of the words within a sentence,
    - Phrase based
        - Comparing similar phrases in both languages and understanding the pattern by doing the same operation a number of times using the n-gram model.
        - This model also had limitations of not being able to consider the order of phrases within a sentence.
    - Syntax based
        - Use of the subject, the predicate, other parts of the sentence, and then to build a sentence tree using language syntax along with rest of word and phrase translation for translation from one language to another.
        - Complexity of model increased with large amount of dataset consisting of numerous sentences.

- Neural Machine Translation(NMT)
    - Recurrent Neural Networks(RNN)
        - RNN are used to reorder the source to target language translation by the help of semi-supervised learning methods.
        - Word embedding is used to generate the vector value of the words which can be used for translation purposes.
        - RNN are based on the idea of binary tree structure for mapping the word in the target language with the vector values of the source language.
        - This model has a complex structure which in turn leads to more number of computations.
    - Long Short-Term Memory Networks(LSTM)
        - LSTM works on the idea of sequence by sequence learning.
        - It uses a NMT model which has eight layers of encoders and decoders within a deep neural network.
        - It is also known as Bidirectional RNN because of the use of encoder and decoder structure.
        - Encoder-Decoder structures are jointly trained to maximize the conditional probabilities of the target sequence given the input sequence.

2.2. Proposed System

- LSTM model
    - This model is a special kind of RNN capable of learning long-term dependencies.
    - The model mainly consists of steps:
        - Decide the information to pass through the cell state
        - Decide what information to store in the cell state
        - Update the cell state by adding new information and discarding the information no longer needed
        - Predict the output based on the information

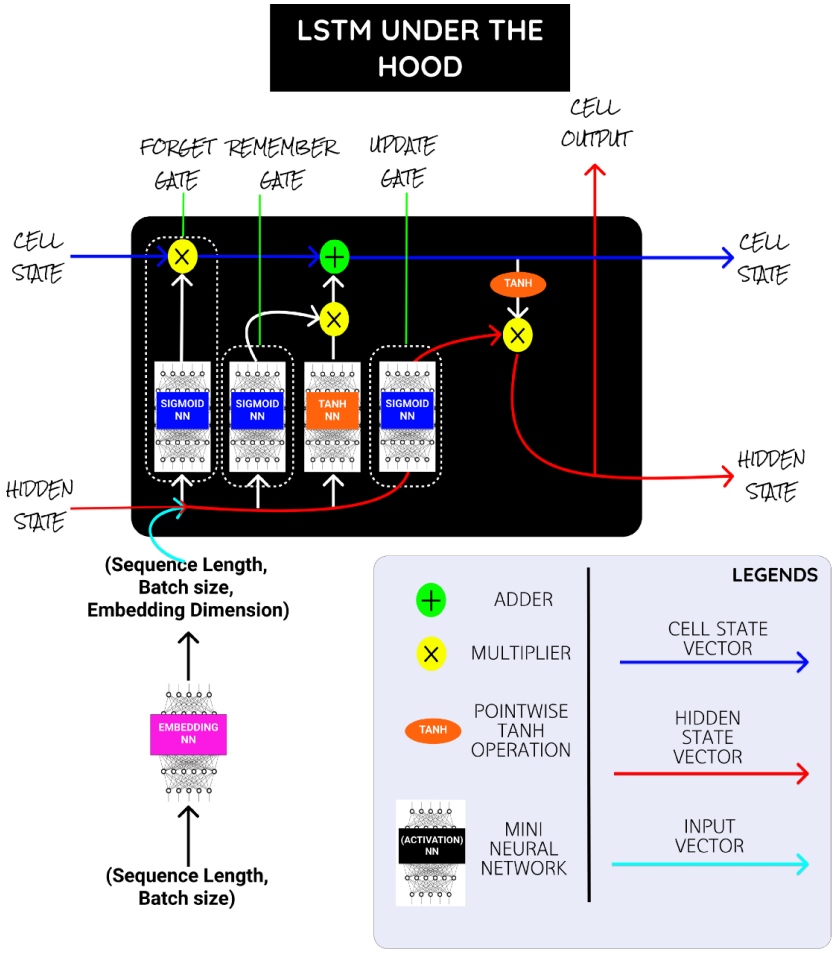

Figure 1.

2.3. Feasibility Study

- Comparison between RNN and LSTM
    - Gated Recurrent Units(GRU) gating values are close to LSTM values with lesser run time in comparison to RNN.
    - The LSTM layer architecture is built in a way such that the network “decides” whether to modify its “internal memory” or not, at each step.
    - Doing so, and if properly trained, then the layer can keep track of important events from further in the past, allowing for much richer inference.
    - LSTM networks were made with the purpose to solve long term dependency problems that traditional RNNs have.
    - LSTM networks are specially made to solve long-term dependency problems in

RNNs and they are super good at making use of data from a while ago (inherent in its nature) by using cell states and RNN has a complex structure which in turn has more computations therefore making LSTM more preferred over RNN.

3. System Analysis & Design

3.1. Aim

To implement a model which can perform word-by-word translation of a given english sentence into hindi with accurate results.

3.2. LSTM Architecture details

- The LSTM reads data one sequence after the other.
- If the input is of length ‘k’ then it will read it ask time-stamps
- The main aim is to bring long and short term dependencies
- The whole system is simplified by 3 Gates and 1 state:
    - Forget Gate: This gate will decide that information should be passed from previous state (h(t-1)) to next state (h(t)) or not
    - Input Gate: This gate deals with the data to update the cell states
    - Output Gate: This decides the information that is relevant to next hidden state
    - Cell state: The pointwise addition of the output of Forget and Input gate updates cell state
- Input tokens are represented by {x1,x2,...,xk}, hidden states by {h1,h2,...,hk}, cell states by {c1,c2,...,c3} and output token by {y1,y2…,yk}.

LSTM consists of 3 neural network architectures.

- Encoder
- Decoder Train Mode
- Decoder Inference Mode

3.2.1. Encoder

- Encoder basically summaries the whole sentence

- The inputs for the encoder are feature vectors of each word of the sentence which is shown in the figure
- After embedding inputs to the encoder, it will generates hidden and cell states at each time and gives final outputs y1, y2, ..yk
- In this project, the hidden and cell states are only stored and outputs are discarded because the states are helpful to store the summarized sequence till the previous state. For this application the states are only needed

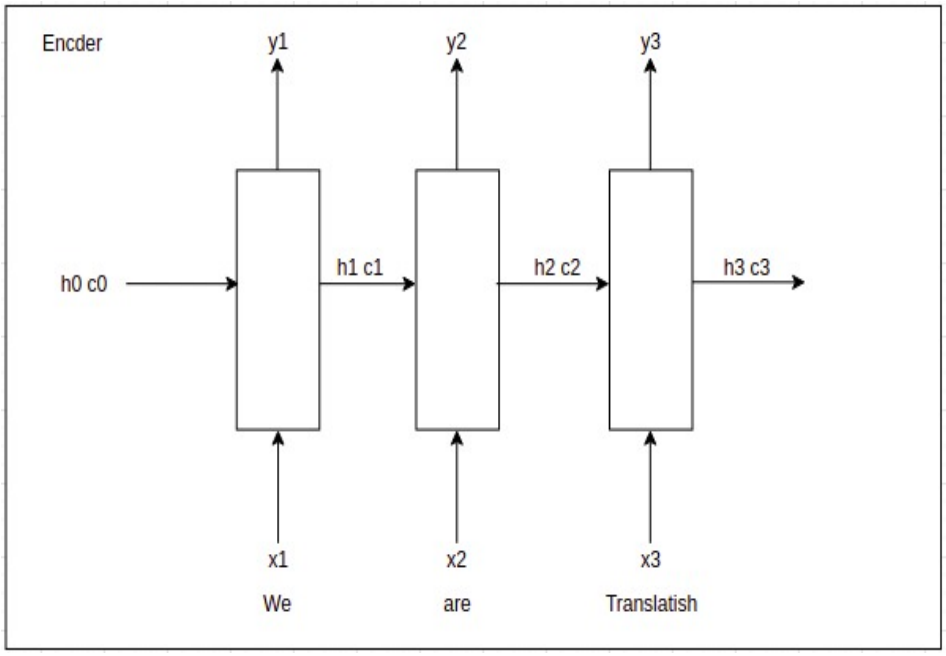

Figure 2.

3.2.2. Decoder

3.2.2.1. Decoder Train Mode

- This accepts the final states of encoder as the input states
- Then getting word start_ and it starts generate word
- At the next instance, We give actual Hindi translation words as input form training labels and it will again predict the next word (Shown in the below figure).
- This uses “Teacher forcing method”

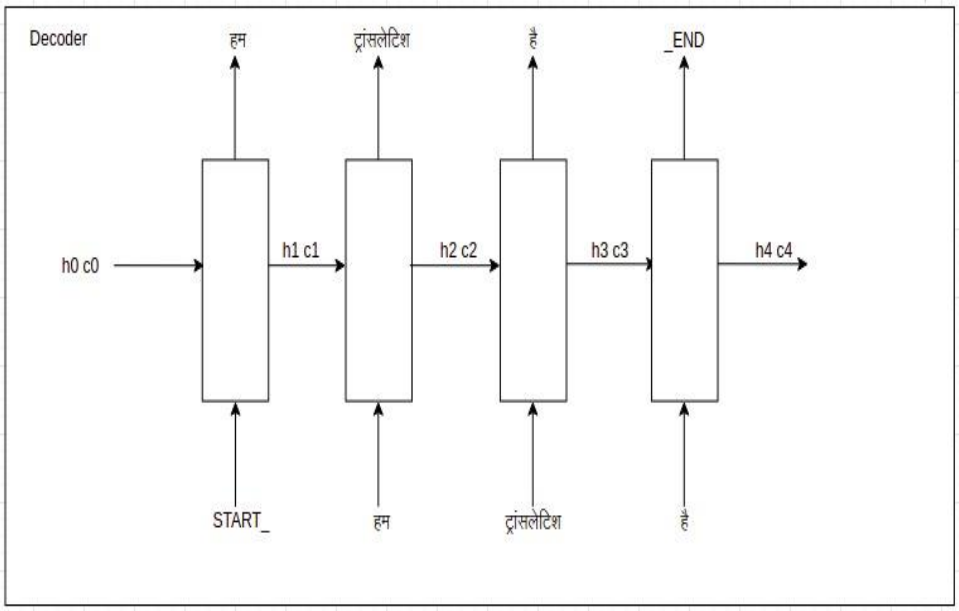

Figure 3.

3.2.2.2. Decoder Inference Mode

- In this application we need slightly different architecture for output testing Decoder.
- As we don't have labels for testing purposes, we need to pass the predicted word to the next timestamp's input (Shown in the below figure).

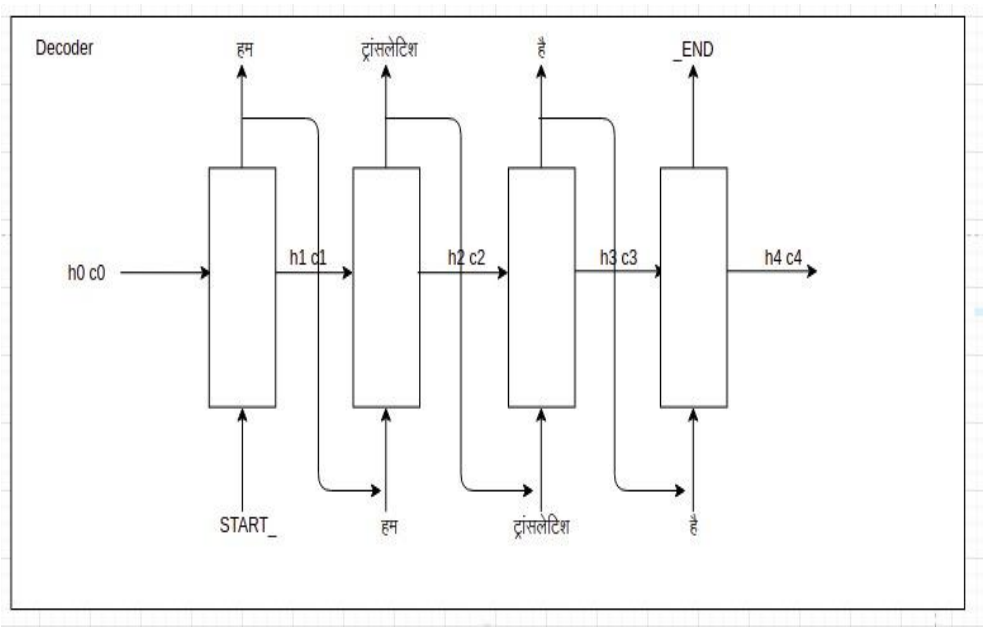

Figure 4.

3.3. Impact of system

- Video marketing can be even more effective overseas.
- Translation helps in cutting across language barriers and interacting with people in such countries.

3.4. Dataset

- Bilingual corpora
- Part of parallel corpora which contains multiple corporas.
- Taken from about 37726 TED talks, news articles, Wikipedia articles.
- Has unique 1,24,318 english records and 9,7662 hindi records
- Corpus link: HindiEnglishCorpora
- Cleaned dataset
- It is sentence tokenized (sentence aligned)

politicians do not have permission to do what needs to be done.	राजनीतिज्ञों के पास जो कार्य करना चाहिए, वह करने कि अनुमति नहीं है .
I'd like to tell you about one such child,	मई आपको ऐसे ही एक बच्चे के बारे में बताना चाहूंगी,

Figure 5.

3.5. Preprocessing

- Removal of duplicate and null records
- Conversion to lowercase
- Removal of Quotes, Special characters, extra spaces
- Computing the length of english and corresponding Hindi sentence, append it in the dataset
- Dictionary of all the words of Hindi and English
- Adding 'START_' and '_END' tokens to the Hindi labels, such that decoder can understand starting and end of the sentences

3.6. Word Embedding

- For feature vector representation, there are many techniques like word embedding, word2vect, etc.
- Here word embedding is used to represent word features and it can be implemented using Keras API in python
- In embedding, there are certain features which are extracted automatically from the text. In which if the number of words is n and the number of features is m then the matrix will be the size of mxn

$$\mathbf{E_{50\times 10K}} = \begin{array}{c} \begin{matrix} \text{Man} & \text{Woman} & \text{King} & \text{Queen} \end{matrix} \\ \begin{bmatrix} -1 & 1 & -0.99 & 0.98 \\ 0.02 & 0.01 & 0.94 & 0.97 \\ \vdots & \vdots & \vdots & \vdots \\ 0.78 & 0.84 & 0.91 & 0.92 \end{bmatrix} \end{array} \begin{matrix} \\ \text{Gender} \\ \text{Royal} \\ \\ \text{Kind} \end{matrix}$$

Figure 6.

- In the above example, n=10K and m=50.
- The features are like gender, royal, kind etc.
- For word which having any particular feature has High value in the matrix
- As King, Queen, Man, Woman all can relate to Gender. So their corresponding values are High but Man is not related to Royal. So that value is near to zero

3.7. Code Snippets(Algorithm and Pseudocode)

- Pre-processing

```
[ ] #Convert to Lowercase
    data['english_sentence'] = data['english_sentence'].apply(lambda x: x.lower())
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda x: x.lower())

    #Removing the Quotes
    data['english_sentence'] = data['english_sentence'].apply(lambda x: re.sub("'", '', x))
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda x: re.sub("'", '', x))

    #special characters
    exclude = set(string.punctuation) # Set of all special characters
    print(exclude)
    data['english_sentence'] = data['english_sentence'].apply(lambda sen : ''.join(char for char in sen if char not in exclude))
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda sen : ''.join(char for char in sen if char not in exclude))

    #removing the extra spaces
    data['english_sentence'] = data['english_sentence'].apply(lambda sen : sen.strip())
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda sen : sen.strip())
    data['english_sentence'] = data['english_sentence'].apply(lambda sen : re.sub(" +", " ", sen))
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda sen : re.sub(" +", " ", sen))

    #adding start and end tokens in hindi sentence
    data['hindi_sentence'] = data['hindi_sentence'].apply(lambda x : 'START_ '+ x + ' _END')

    # Dictionary mapping for all unique words
    all_eng_words=set()
    for eng in data['english_sentence']:
        for word in eng.split():
            if word not in all_eng_words:
                all_eng_words.add(word)
    all_hindi_words=set()
    for hin in data['hindi_sentence']:
        for word in hin.split():
            if word not in all_hindi_words:
                all_hindi_words.add(word)

    input_words = sorted(list(all_eng_words))
    target_words = sorted(list(all_hindi_words))
    input_token_index = dict([(word, i+1) for i, word in enumerate(input_words)])
    target_token_index = dict([(word, i+1) for i, word in enumerate(target_words)])
```

Figure 7.

- Batch Generation:

Algorithm 1 Batch Generation

Input: X-train, Y-train, batch-size

```
While(1)
    for j ← 0 to batch-size-1
        initialize encoder-input-data with zero(batch-size x max-len-eng)
        initialize decoder-input-data with zero(batch-size x max-len-eng)
            for i, input, target in one batch
                for t, word in input
                    encoder-input-data[i,t] ← dict[word] (index in global dict)
                end for
                for t, word in target
                    decoder-input-data[i,t] ← dict[word] (index in global dict)
                end for
            end for
        pass one batch [encoder-input-data, decoder-input-data]
    end for
```

Figure 8.

```
[ ]  def generate_batch(X = X_train, y = y_train, batch_size = 128):
         ''' Generate a batch of data '''
         while True:
             for j in range(0, len(X), batch_size):
                 encoder_input_data = np.zeros((batch_size, max_length_src),dtype='float32')
                 decoder_input_data = np.zeros((batch_size, max_length_tar),dtype='float32')
                 decoder_target_data = np.zeros((batch_size, max_length_tar, num_decoder_tokens),dtype='float32')
                 #print("ok")
                 for i, (input_text, target_text) in enumerate(zip(X[j:j+batch_size], y[j:j+batch_size])):
                     #print(i)
                     for t, word in enumerate(input_text.split()):
                         encoder_input_data[i, t] = input_token_index[word] # encoder input seq
                     for t, word in enumerate(target_text.split()):
                         if t<len(target_text.split())-1:
                             decoder_input_data[i, t] = target_token_index[word] # decoder input seq
                         if t>0:

                             decoder_target_data[i, t - 1, target_token_index[word]] = 1.
                 yield([encoder_input_data, decoder_input_data], decoder_target_data)
```

Figure 9.

- Model Building

```
[ ]  # Encoder
     encoder_inputs = Input(shape=(None,))
     enc_emb =  Embedding(num_encoder_tokens,latent_dim, mask_zero = True)(encoder_inputs)
     encoder_lstm = LSTM(latent_dim, return_state=True)
     encoder_outputs, state_h, state_c = encoder_lstm(enc_emb)
     # We discard `encoder_outputs` and only keep the states.
     encoder_states = [state_h, state_c]
```

```
[ ]  # Decoder
     decoder_inputs = Input(shape=(None,))
     dec_emb_layer = Embedding(num_decoder_tokens, latent_dim, mask_zero = True)
     dec_emb = dec_emb_layer(decoder_inputs)

     decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
     decoder_outputs, _, _ = decoder_lstm(dec_emb,
                                          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"])
```

Figure 10.

- Model Summary

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, None)]	0	
input_4 (InputLayer)	[(None, None)]	0	
embedding_2 (Embedding)	(None, None, 256)	4718848	input_3[0][0]
embedding_3 (Embedding)	(None, None, 256)	5631488	input_4[0][0]
lstm_2 (LSTM)	[(None, 256), (None,	525312	embedding_2[0][0]
lstm_3 (LSTM)	[(None, None, 256),	525312	embedding_3[0][0] lstm_2[0][1] lstm_2[0][2]
dense_1 (Dense)	(None, None, 21998)	5653486	lstm_3[0][0]

Total params: 17,054,446
Trainable params: 17,054,446
Non-trainable params: 0

Figure 11.

- BLEU score computation
    - For n-gram precision calculation:

Algorithm 2 n-gram precision calculation

Input: n

Output: Score

1: Compute n-gram pair out of the predicted sentence

2: Make table of unique pair and its count of occurrence in the predicted sentence

3: total-count ← Number of pairs of n-grams

4: count ← $\sum$ number of times particular n-gram is in the target sentence

5: score ← count/total-count

Figure 12.

- Cumulative n-gram score:

Algorithm 3 Cumulative n-gram score

Input:weight-1,weight2,...weight-n

Output:Final-score

```
Score empty list
    for i → 1 to n
        Score[i] ← BLEUscore(i)
    end for
Final-score = 0
    for i → 1 to n
        Final-score = Final-score + weight-i * Score[i]
    end for
```

Figure 13.

```
[ ]  import nltk

     score=0
     s_list=[]
     for i in range(len(X_test)):
         #print('Input English sentence:', X_test[i:i+1].values[0])
         #print('Actual Hindi Translation:', y_test[i:i+1].values[0][6:-4])

         (input_seq, actual_output), _ = next(train_gen)

         decoded_sentence = decode_sequence(input_seq)
         BLEUscore = nltk.translate.bleu_score.sentence_bleu([y_test[i:i+1].values[0][6:-4]], decoded_sentence[:-4], weights = (0.5, 0.5))
         score+=BLEUscore
         print(i)
         s_list.append(BLEUscore)
         #print(BLEUscore)
         #print('Predicted Hindi Translation:', decoded_sentence[:-4])
         #print('------------------------------')
         #print()

         #if i>10:
         #    break
```

Figure 14.

3.8. Testing

For testing the applications like Neural Machine Translation, BLEU algorithm is used

BLEU score:

- Bilingual evaluation understudy
- Number between 0-1
- The number indicates how much similar the predicted text is to reference text
- 1 indicates the maximum similarity
- It can be using Unigram, bigram,... or the combination of any(Here 2 and 4-gram cumulatives are used)
- The algorithm is mentioned above

Example: **Reference 1**: The cat is on the mat.

Reference 2: There is a cat on the mat.

Predicted Output: The cat the cat on the mat. (Target)

Bigram BLEU Score Computation

Possible Bigrams	**Frequency in Target Sentence**	**Actual frequency from the reference sentences**
the cat	2	1
cat the	1	0
cat on	1	1
on the	1	1
the mat	1	1

Table 1.

- Our total count is 2+1+1+1+1=6
- Actual count is 1+0+1+1+1=4
- Bleu score = count/total = 4/6

3.9. System Deployment

As the system is divided into steps:

- Uploading the video in English language
- Video to audio conversion
- Audio to text conversion
- English text to Hindi translation
- Displaying the Translation

4. Results

4.1. Outputs

- Preprocessing Results

	source	english_sentence	hindi_sentence	english_sen_length	hindi_sen_length
78002	ted	other creatures like the crows	START_ दूसर जीव जैसे कि कौए _END	5	7
63794	ted	a degree of environmental awareness for instance	START_ जैसे थोड़ी बहुत पर्यावरण सचेतनता _END	7	7
66909	indic2012	douliganga merge with algunada and unites in vishnuprayag	START_ धौली गंगा का अलकनंदा से विष्णु प्रयाग में संगम होता है। _END	8	13
61994	ted	americans have the voice of america and the fulbright scholarships	START_ अमरीकनों के पास अमरीका की आवाज़voice of america और फुल्ब्राईट छात्रवृत्तिfulbright scholarships है _END	10	15
41525	indic2012	janpath samachar major hindi daily newspaper of north east india	START_ जनपथ समाचार पूर्वोत्तर भारत का प्रमुख हिन्दी दैनिक _END	10	10
51597	tides	any problems	START_ कोई समस्या _END	2	4
56131	ted	you need a multimilliondollar plant like this	START_ करोड़ों की लागत वाली ऐसे प्लान्ट की ज़रूरत थी _END	7	11
26396	ted	and perhaps they knew the future also	START_ और संभवतः वे भविष्य भी जानते थे। _END	7	9
56248	ted	and so you can start reading on your ipad in your living room	START_ और तो आप इसे एक कमरे में अपने आई पैड पर पढना शुरू कर सकते हैं _END	13	18
105530	ted	and talked and shared our difficulties	START_ और अपनी कठिनाइयो के बारे मे बात की है और उन्हे बांटा है _END	6	15

Figure 15.

- Model Training
    - After training the model for 50 epochs, it starts to converge which suggests that the training should be done till 50 epochs.

```
Epoch 39/50
619/619 [==============================] - 329s 531ms/step - loss: 0.9872 - acc: 0.8767 - val_loss: 1.9284 - val_acc: 0.7631
Epoch 40/50
619/619 [==============================] - 329s 531ms/step - loss: 0.9734 - acc: 0.8785 - val_loss: 1.9110 - val_acc: 0.7660
Epoch 41/50
619/619 [==============================] - 329s 531ms/step - loss: 0.9596 - acc: 0.8807 - val_loss: 1.9069 - val_acc: 0.7667
Epoch 42/50
619/619 [==============================] - 329s 531ms/step - loss: 0.9470 - acc: 0.8824 - val_loss: 1.8915 - val_acc: 0.7683
Epoch 43/50
619/619 [==============================] - 329s 532ms/step - loss: 0.9361 - acc: 0.8840 - val_loss: 1.8818 - val_acc: 0.7705
Epoch 44/50
619/619 [==============================] - 329s 532ms/step - loss: 0.9237 - acc: 0.8858 - val_loss: 1.8723 - val_acc: 0.7728
Epoch 45/50
619/619 [==============================] - 329s 532ms/step - loss: 0.9128 - acc: 0.8873 - val_loss: 1.8627 - val_acc: 0.7745
Epoch 46/50
619/619 [==============================] - 328s 530ms/step - loss: 0.9029 - acc: 0.8886 - val_loss: 1.8491 - val_acc: 0.7769
Epoch 47/50
619/619 [==============================] - 330s 534ms/step - loss: 0.8931 - acc: 0.8903 - val_loss: 1.8442 - val_acc: 0.7766
Epoch 48/50
619/619 [==============================] - 328s 530ms/step - loss: 0.8840 - acc: 0.8914 - val_loss: 1.8374 - val_acc: 0.7789
Epoch 49/50
619/619 [==============================] - 329s 531ms/step - loss: 0.8753 - acc: 0.8928 - val_loss: 1.8333 - val_acc: 0.7801
Epoch 50/50
619/619 [==============================] - 330s 533ms/step - loss: 0.8675 - acc: 0.8937 - val_loss: 1.8226 - val_acc: 0.7806
```

+ Code + Markdown

Figure 16.

- Model Validation
    - After 30 epochs the model stops learning which suggests to stop the training of the model at 30 epochs.

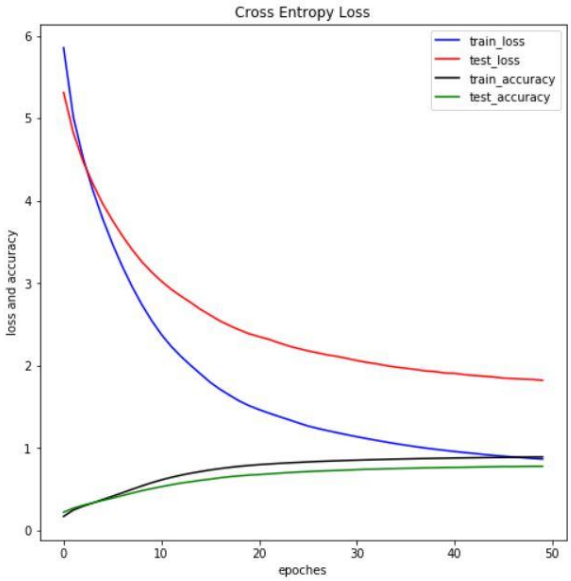

Figure 17.

- Model Testing

```
Input English sentence: she went to school to get the information
Actual Hindi Translation:  वो जानकारी लेने विद्यालय जाती थी
Predicted Hindi Translation:  वो जानकारी लेने विद्यालय जाती थी
```

Figure 18.

Example1

Here, the model is able to predict the hindi translation for the given english sentence exactly similar to the actual hindi translation.

```
Input English sentence: it doesnt matter if you fail
Actual Hindi Translation:  कोई फ़र्क नहीं पडता यदि आप फ़ेल हो गये तो।
Predicted Hindi Translation:  कोई फ़र्क नहीं पड़ता कि यदि आप लोगों को छोड़ रहे
```

Figure 19.

Example2

Here, the model is able to predict the hindi translation for the given hindi translation with a bit difference with the actual hindi translation due to difference in semantics and syntactics of the sentence

- Bleu Score

Sr. no.	Dataset Name	Description	BLEU Score
1.	HindiEnglishCorpora	Contains TEDtalks, news articles and Wikipedia articles	53.01%
2.	Machine-Translation-English-To-Hindi	Contains day-to-day routinely used sentences	36.9%
3.	IITBombay English-Hindi Corpus Dataset	Contains Indian judicial statements and their corresponding translated sentences	12%

Table 2.

4.2. Working end Product (Web app)

4.2.1. Technology Used

4.2.1.1. Frontend

- HTML
- CSS
- React JS

4.2.1.2. Backend

- Django
- Django Rest-frameworks
- Python
- Tensorflow
- Keras
- Audiopy, pydub

4.2.2. SnapShots of Web App

- Welcome (Intro) Screen **:**

This is the intro screen when you reach out to our web app.

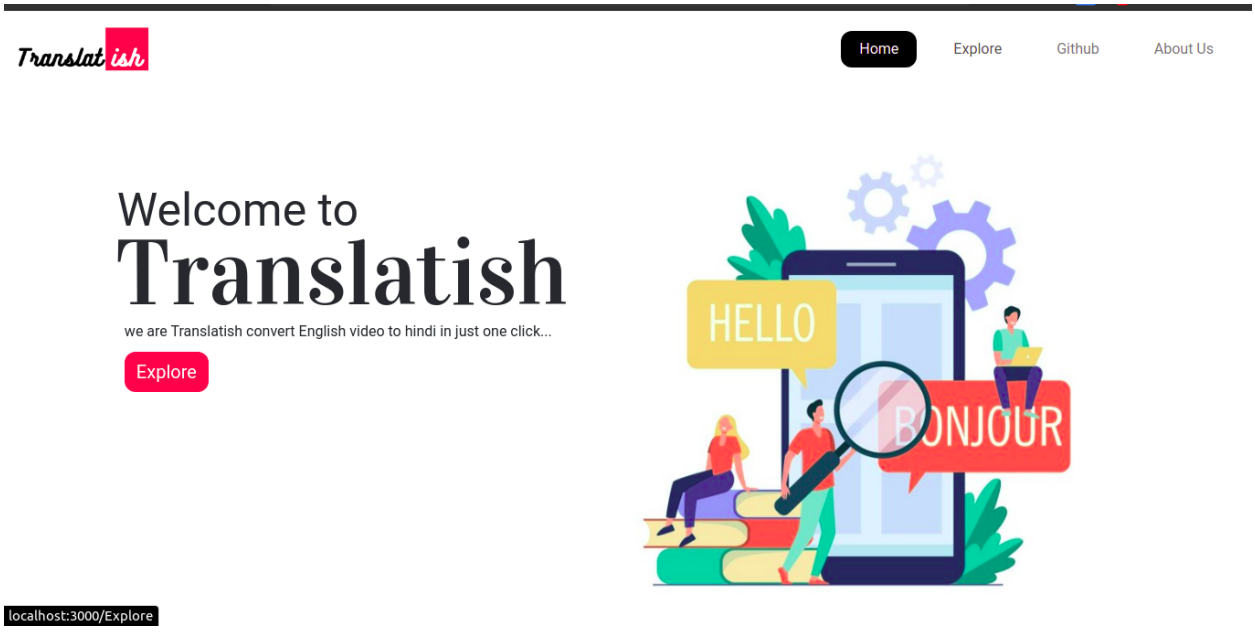

Figure 20.

- About Us**:**

About us, the screen includes the details of the project member with their profile pictures. (Meet the Team)

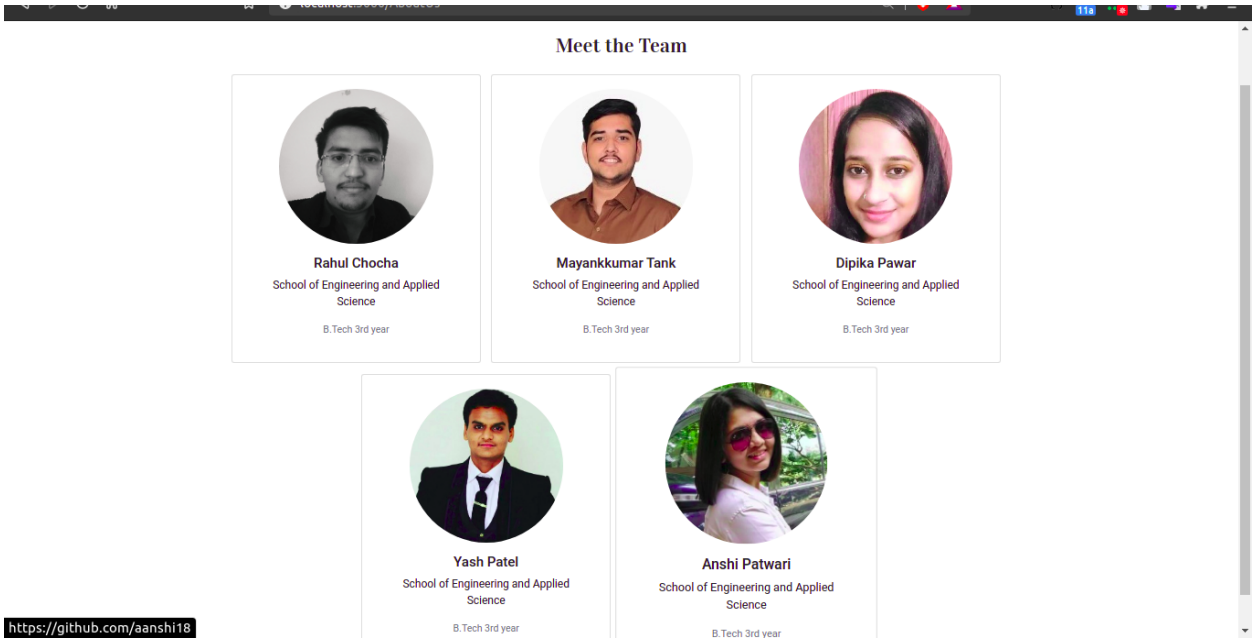

Figure 21.

- Explore Page**:**

This is the main screen where you can do your desired work (video translation). First snapshot showing the page without uploading the video. There you can see that there is an option of uploading the video with click or drag and drop facility. Talking about the second snapshot where video is uploaded and there you can play the video. And at the end of the page you can see two divisions one shows the extracted english sentences (converted from the video). And Another one shows the translated hindi text of the video.

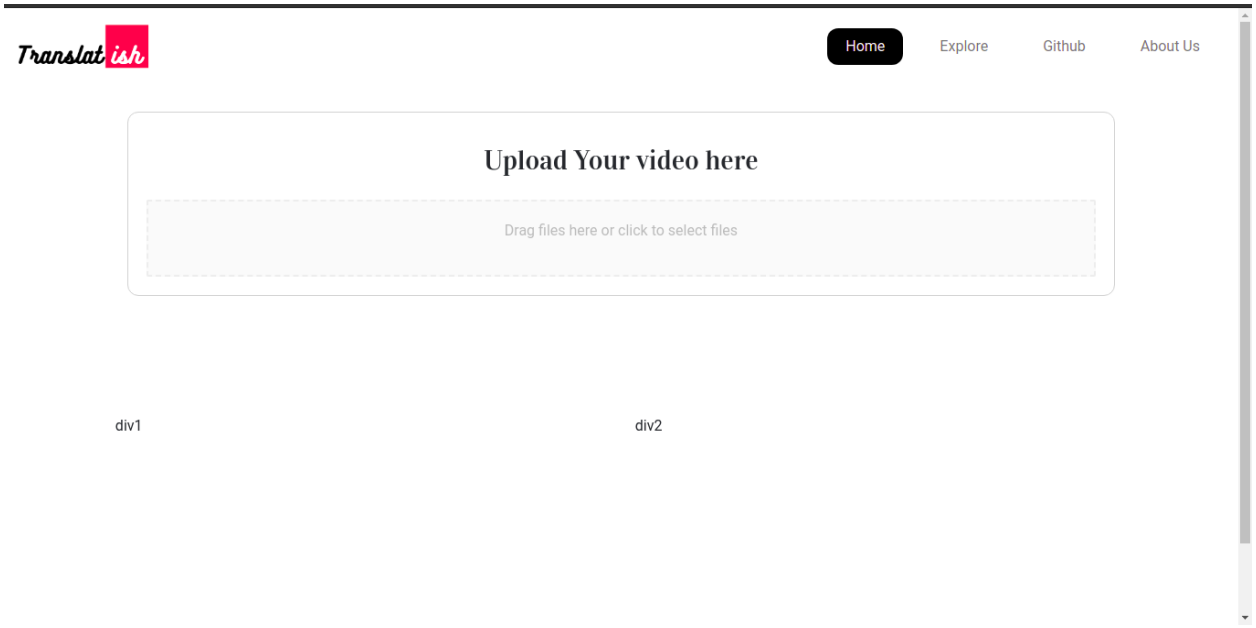

Figure 22.
Snapshot 1

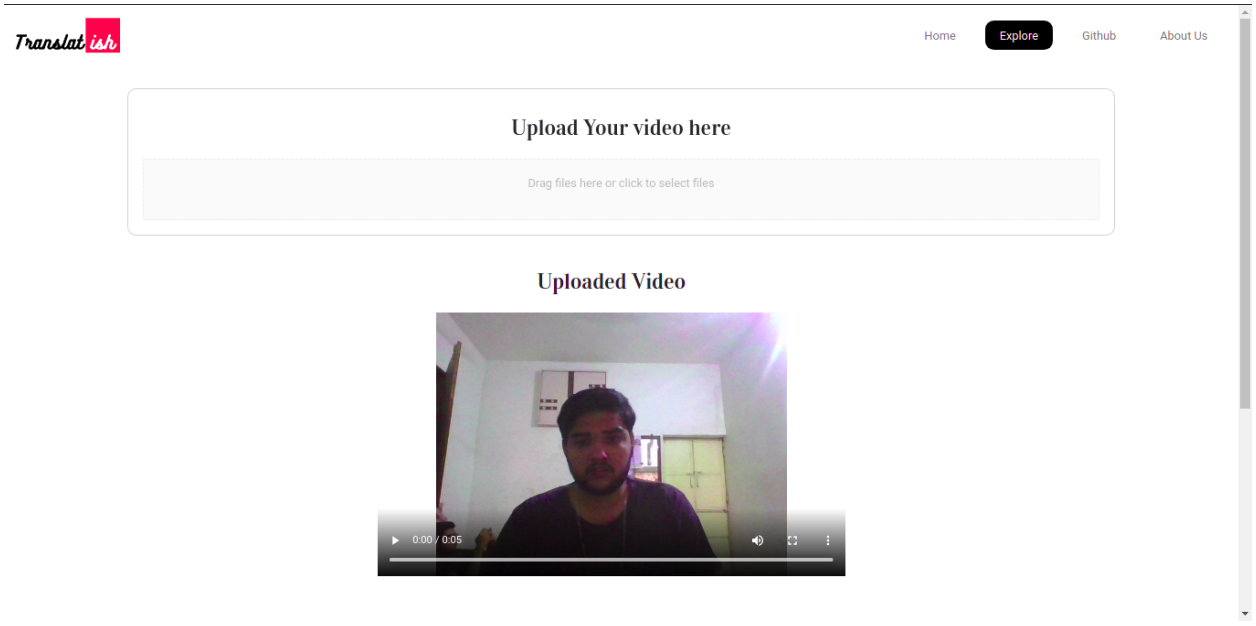

Figure 23.
Snapshot 2

5. Conclusion and Future Work

5.1. Conclusion

- The implementation of LSTM model for the purpose of video translation i.e. neural machine translation gave accurate results for videos similar to the training dataset
- There was a bit of fluctuation in the results for translation of videos different from the training dataset type.

5.2. Future Work

- Implementation of proposed model on different datasets
- Try out translation into other languages
- Train the model on dataset for variety of sentences from different videos
- Conversion of the translated sentences into video format
- Improve the model performance by implementing it on super computers which can provide more efficient results and computation power
- Change the speech to text conversion strategy.

6. References

6.1. Literature Review

[1] Translartisan. (2018, August 31). Rule-based machine translation. Retrieved from https://translartisan.wordpress.com/tag/rule-based-machine-translation/

[2] A., P., & P. (2018). The IIT Bombay English-Hindi Parallel Corpus. 1-4. Retrieved May 19, 2018, from https://arxiv.org/pdf/1710.02855.pdf.

[3] Saini, S., & Sahula, V. (2018). Neural Machine Translation for English to Hindi. *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*. doi:10.1109/infrkm.2018.8464781

[4] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi and S. Jain, "Machine translation using deep learning: An overview," 2017 International Conference on Computer,

Communications and Electronics (Comptelix), 2017, pp. 162-167, doi: 10.1109/COMPTELIX.2017.8003957.

6.2. LSTM Architecture

[5] Lamba, H. (2019, February 17). Word Level English to Marathi Neural Machine Translation using Seq2Seq Encoder-Decoder LSTM Model. Retrieved from https://towardsdatascience.com/word-level-english-to-marathi-neural-machine-translation-using-seq2seq-encoder-decoder-lstm-model-1a913f2dc4a7

[6] Ranjan, R. (2020, January 16). Neural Machine Translation for Hindi-English: Sequence to sequence learning. Retrieved from https://medium.com/analytics-vidhya/neural-machine-translation-for-hindi-english-sequence-to-sequence-learning-1298655e334a

[7] V, B. (2020, November 16). A Comprehensive Guide to Neural Machine Translation using Seq2Sequence Modelling using PyTorch. Retrieved from https://towardsdatascience.com/a-comprehensive-guide-to-neural-machine-translation-using-seq2sequence-modelling-using-pytorch-41c9b84ba350

[8] Understanding LSTM Networks. (n.d.). Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[9] Pedamallu, H. (2020, November 30). RNN vs GRU vs LSTM. Retrieved from https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573

[10] Mittal, A. (2019, October 12). Understanding RNN and LSTM. Retrieved from https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

[11] Culurciello, E. (2019, January 10). The fall of RNN / LSTM. Retrieved from https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0