Translitorator -
Recurrent Neural Network For Machine Translation

Submitted by,

Name	Registration Number
Samridhi Murarka	17BCE2038
Manish Jain	17BCE0819
Tejas Jambhale	17BCE0861

A Report submitted for
the Final Project Review of

Natural Language Processing
CSE3013
Slot: A2 + TA2
Winter Semester 2019-20

Under the guidance of

Professor: Sharmila Banu K.
(School Of Computer Science And Engineering)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

TABLE OF CONTENTS

PROBLEM STATEMENT

Speech translation innovation—having the option to talk and have one's words made an interpretation naturally into the other individual's language—has for quite some time been a fantasy of mankind. Speech translation is basically a way to convert the language spoken by a certain individual into another preferable language. We aim to focus on a three-stage model for this very idea. The three stages include: *Speech Recognition, Translation and Synthesis.* These are connected end to end.

This project aims to create a model that is able to facilitate speech translation among various languages using a three-stage model. These three main modules, Speech Recognition, Machine Translation and Speech Synthesis use different tools and concepts for their specific purpose. The need of Google API for converting speech to text is discussed. The use of RNN for helping in translation is elaborated and further the unique part of speech synthesis is explained in detail. In addition, we aim to provide a system architecture, services and communication protocols for connecting the client to the main speech to speech translation servers. We can summarize the whole project as a system to facilitate machine translation using only a recursive neural network.

LITERATURE REVIEW

The paper *Review of speech-to-text recognition technology for enhancing learning* discusses the effectiveness of STR application on students learning performance, during and after the collaborative learning activities on the online synchronous cyber classrooms. The application has been implemented on the open university students, using STR. The usage of STR was implemented among students with cognitive environment, collaborative academic activities, non-native speakers and students. Dataset used in this paper is Windows Speech Recognition in the Microsoft Operating System, IBM ViaVoice software. The paper has used its dataset to be the Windows Speech Recognition in the Microsoft Operating System IBM ViaVoice software. Most experimental students perceived that STR was useful for individual presentations and for essays writing. Students who were exposed to the STR were willing to use STR system for learning in the future. One of the most common concerns reported in relation to online learning literature is the poor audio quality due to restricted internet bandwidth availability. The above-mentioned problems can be solved by adopting some assistive media-to-text recognition technologies, such as writing-to-text, image-to-text, diagram-to-text, text-to-speech, speech-to-text, and handwriting-to-text. (Shadiev, R., Hwang, W. Y., Chen, N. S., & Huang, Y. M., 2014) [1]

The paper *Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition* deals with usage of deep bi-directional LSTM RNNs with CTC loss. Neural Speech Recognizer model has a deep LSTM RNN architecture built by stacking multiple LSTM layers. Since the bidirectional RNN models have better accuracy and their application is offline speech recognition on the system, they have used two LSTM layers at each depth - one operating in the forward and another operating in the backward direction in time over the input sequence. Vocabulary corpus has been used as dataset here, that has 296 videos from 13 categories, with each video averaging 5 minutes in length. The total test set duration is roughly 25 hours and 250,000 words. The output from the CTC layer, essentially making the CTC word model an end-to-end all-neural speech recognition model. The bi-directional LSTM CTC word models are capable of accurate speech recognition with no language model or decoding involved. The error rate calculation disadvantages the CTC spoken word model as the references are in written domain, but the output of the model is in spoken domain, creating artificial errors like "three" vs "3". The entire speech recognizer becomes a single neural network. (Soltau, H., Liao, H., & Sak, H., 2016) [2]

According to *Achieving human parity in conversational speech recognition*, the use of various convolutional and LSTM acoustic model architectures, combined with a novel spatial smoothing method and lattice-free MMI acoustic training, multiple recurrent neural network language modeling approaches, and a systematic use of system combination. They have used CNN, LSTM, Spatial smoothing, speaker adaptive modelling, lattice-free sequence training. The 4-gram language model for decoding was trained on the available CTS transcripts from the DARPA EARS program: Switchboard (3M words), BBN Switchboard-2 transcripts (850k), Fisher (21M), English CallHome (200k), and the University of Washington conversational Web corpus (191M). The only exception is the VGG+ResNet system, which combines acoustic senone posteriors from the VGG and ResNet networks. While this yields our single best acoustic model, only the individual VGG and ResNet models are used in the overall system combination. In the included model, N-best output from all systems are combined confusion network construction generates new possible hypotheses not contained in the original N-best lists the machine errors are substantially the same as human ones, with one large exception: confusions between backchannel words and hesitations. (Xiong, W., Droppo, J., Huang, X., Seide, F., ... & Zweig, G., 2016) [3]

Neural responding machine for short-text conversation discusses a general encoder-decoder framework: it formalizes the generation of response as a decoding process based on the latent representation of the input text, while both encoding and decoding are realized with recurrent neural networks (RNN). The dataset consists of a corpus of roughly 4.4 million pairs of conversations from Weibo. They use an encoder-decoder based neural network to generate a response in STC. They have also empirically verified that the proposed method can yield performance better than traditional retrieval-based and translation-based methods. Widely accepted evaluation methods in translation do not apply. It is also not reasonable to evaluate with Perplexity, a generally used measurement in statistical language modeling, because the naturalness of response and the relatedness to post cannot be well evaluated. Natural language conversation is one of the most challenging artificial intelligence problems, which involves language understanding, reasoning, and the utilization of common-sense knowledge. Automatic evaluation of response generation is still an open problem. (Shang, L., Lu, Z., & Li, H., 2015) [4]

This paper compares the performances of different recurrent neural networks (RNNs) units, particularly a long short-term memory unit (LSTM) and a gated recurrent unit (GRU), in sequence-to-sequence models, to see if different types of recurrent units can have biases in the performance of translating certain language pairs, as well as translation directions. Methods used in this paper are recurrent neural network RNN, Long Short-Term Memory Unit which are applied on French-English and German-English bidirectional pair corpus. The research paper points out that medium-length sentences showed that they have higher BLEU scores than others and all buckets were able to improve their overall scores with dropout. The overall comparison of the paper is good but all translation experiments were conducted on just two language pairs, French-English and German-English. Also, they could have performed dropout experiment to all eight baseline models or tweaked the dropout probability parameter, but the inability of the paper to do so makes it an insufficient comparison. (Shi Fan, Yili Yu,) [5]

The research work that has been carried out in *Exploring neural transducers for end-to-end speech recognition* discusses empirical comparison among the CTC, RNN-Transducer, and attention-based Seq2Seq models for end-to-end speech recognition. Simplifying speech recognition pipeline so that decoding can be expressed purely as neural network operations. On simplifying speech recognition pipeline so that decoding can be expressed purely as neural network operations. It has been done on Hub5'00 dataset. The choice of the encoder plays a crucial role in optimizing the performance of three models: CTC, RNN-transducer and attention-based seq-seq model. In attempt to train RNN-Transducer models with the streaming constraint, and in reducing computation in encoder layers, it is found that CTC and attention models still have strengths that we aim to leverage in the future work with RNN-Transducers. (Battenberg, E., Chen, J., Child, R., Coates, A., Li, Y. G. Y., ... & Zhu, Z., 2017) [6]

Most of the techniques for machine translations require a large parallel corpora of the language pairs. Getting such a large corpora has become one of the major practical problems. There have also been several techniques like traingulationa and semi-supervised learning techniques but these still require strong cross-lingual signals. This research paper provides us with an unsupervised neural machine translation technique that requires nothing but a monolingual corpora,thereby completely eliminating the earlier faced problems. The custom model used in the paper is based on the unsupervised embedding mappings(RNN-embedded) and modified RNN-encoder decoder model that use a mixture of denoising and back-translation. The model proposed in the paper results in 15.56 BLEU points while translating from French to English and 10.21 BLEU points while translating from German to English. The technique proposed in this paper can also be used for small parallel-corpora. The implementation of the paper has been made open-source by the authors. The excessive constraints during the training of the model might have played a role in the BLEU points reduction. The constraints seem to be very complex and the paper later mentions that reducing the complexity and quantity of these constraints is one of its future aim. They also aim to analyse the denoising technique used by them on several different pairs of languages and explore its different functions.[7]

The paper is inspired by the extended application of neural machine translations(NMT) to multilinguality i.e. being capable of doing multi-directional translation using a single system. These multilingual NMTs have showed quite good results as compared to the strictly bilingual systems and tend to work more effectively and efficiently in low-resource settings.It also mentions how these NMTs provide us with zero-shot systems.The paper basically provides us with: A quantitative and comparative analysis of the translations produced by bilingual, multilingual and zero-shot systems, it provides us with a critical analysis of the two major and trending techniques in machine translation, which are the Recurrent and the Transformer, it closely observes and points out how the closeness between languages affects the zero-shot translation. The analysis in this paper leverages multiple professional post-edits of automatic translations by several different systems and emphasises majorly on the automatic standard metrics (BLEU and TER) and also on the widely used error categories like lexical,morphology and order errors. The paper compares and analyses the bilingual, multilingual and zero-shot systems brilliantly with respect to the errors like lexical,morphological and order errors but it fails to give us a comparison between these different kinds of errors for a certain system (Recurrent or Transformer). The paper managed to give a few differences between these errors but cannot be considered comprehensive enough. [8]

DATASET USED/TOOLS USED

3.1 Where is the dataset taken from?

We will be using a combination of 2 datasets that are converted into two text files. One contains English sentences and the other contains the French translation of each sentence in the latter file. Also, there are approximately 1.4 lakh sentences for us to use in our dataset.

Link to the dataset – http://www.manythings.org/anki/
https://github.com/susanli2016/NLP-with-Python/tree/master/data

3.2 Is your project based on any other reference project?

There are several implementations of text translation such as:

- Google Translator
- Yandex Translate
- iTranslate

3.3 How does your project differ from the reference project?

Traditional apps/tools use ANN for translation but our project does it differently by using RNN(Recurrent Neural Network). Also, our project will be able to listen to the speech, convert it into text, translate the text and talk on behalf of us.

MODULES / HARWARE SOFTWARE REQUIREMENTS

Our project will be divided into three different modules defined below.

4.1 Module 1

This module consists of converting *“what we speak”* into text. To do this we will be using the Google Speech Recognition API.

- We will install the SpeechRecognition library
- Install Pyaudio Package to help access the microphone
- Now we use the inbuilt functions in SpeechRecognition Library to get rid of the ambient noise like people talking in background, construction noise in the background, footsteps,etc in the recorded speech by the user.
- Convert the cleaned speech for conversion and get the text.

4.2 Module 2

This module consists of converting the text found out from the previous module into a different language. We will use the deep neural networks to develop a model based on RNN (Recurrent Neural Network) to do the required translation. We will be using a dataset to train the model and then predict the output of the text. We will also be using 4 different models of RNN:

- Basic RNN
- Recurrent Neural Network with Embedding
- Bidirectional Recurrent Neural Network
- Encoder - Decoder Recurrent Neural Network

We will find out which among the above models gives the best accuracy for a particular language conversion and use it for our *“text translation”*.

4.3 Module 3

This module is where the translated text (i.e. the final text that we got after processing the speech of the user and converting it into the required language) has to be converted into speech again. This is where the computer will speak on our behalf in the specified language. For this again we will be using Google Text to Speech API. It enables us to convert text of several different languages like french,english,german,etc.

4.4 Hardware Requirements

There are no specific hardware requirements other than the most basic things like:

- Laptop
- Microphone
- Speaker
- GPU (required for RNN)

4.5 Software Requirements

We will require a few libraries and a few platforms to run our project.

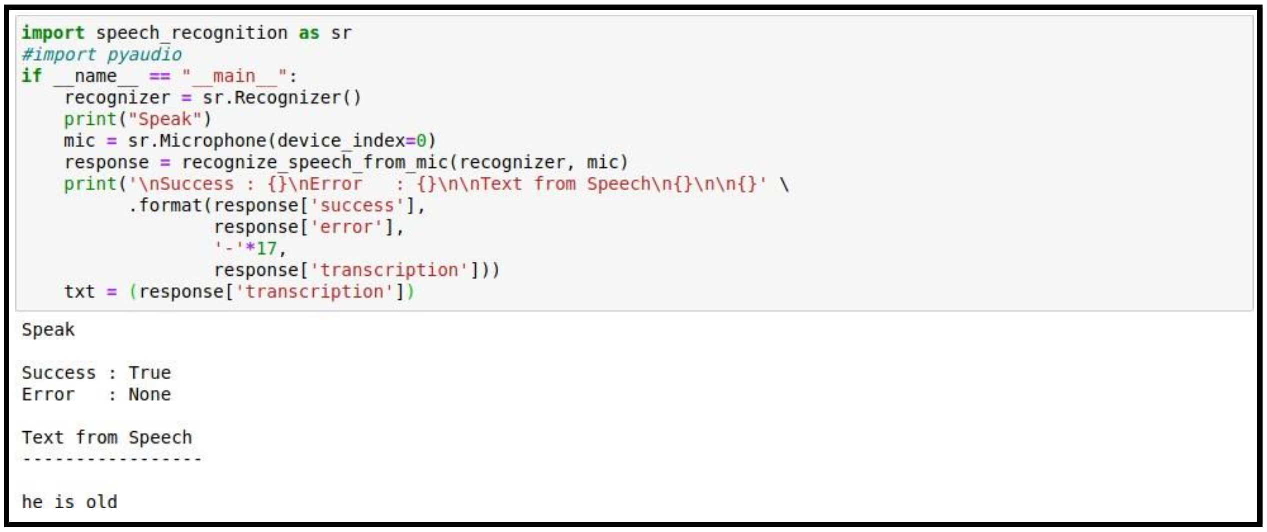
- Spyder (Python 3.7)
- Jupyter
- Libraries like SpeechRecognition, Pyaudio, numpy, tests, helper,etc

APIs like Google Speech Recognition and Google Text to Speech API (gTTS).

IMPLEMENTATION

5.1 Converting speech to text

In this module we are going to convert speech into text using SpeechRecognition by Google and PyAudio to access the microphone on your device. The converted text is then stored in a variable so that it can be used later for translation when we load the Final model.



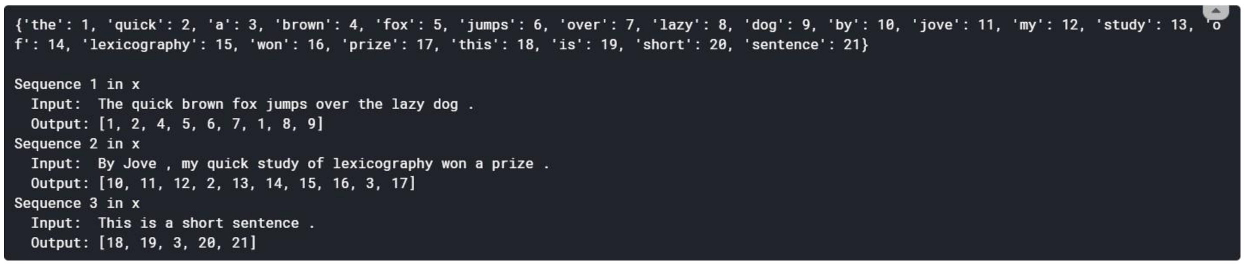
```
import speech_recognition as sr
#import pyaudio
if __name__ == "__main__":
    recognizer = sr.Recognizer()
    print("Speak")
    mic = sr.Microphone(device_index=0)
    response = recognize_speech_from_mic(recognizer, mic)
    print('\nSuccess : {}\nError   : {}\n\nText from Speech\n{}\n\n{}' \
          .format(response['success'],
                  response['error'],
                  '-'*17,
                  response['transcription']))
    txt = (response['transcription'])
```

```
Speak

Success : True
Error   : None

Text from Speech
-----------------

he is old
```

5.2 Module 2

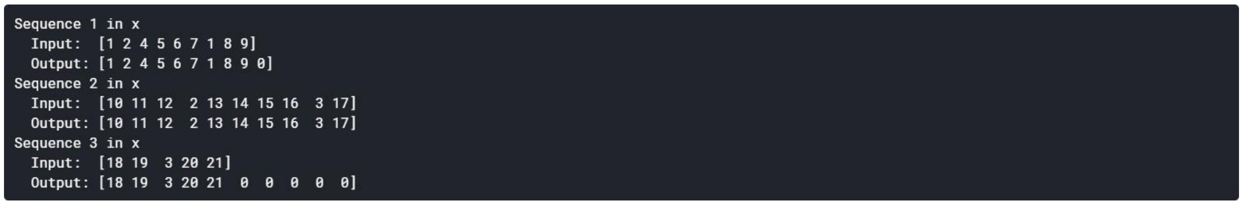
5.2.1 Analysing and Preprocessing the dataset

We will convert all the sentences in our dataset into sentences of equal lengths. To do this we will first tokenize all the sentences using keras tokenizer and then add padding. The keras tokenizer converts/maps all the unique words to a specific integer.



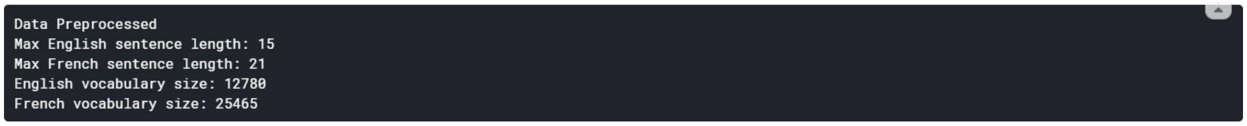
```
{'the': 1, 'quick': 2, 'a': 3, 'brown': 4, 'fox': 5, 'jumps': 6, 'over': 7, 'lazy': 8, 'dog': 9, 'by': 10, 'jove': 11, 'my': 12, 'study': 13, 'o
f': 14, 'lexicography': 15, 'won': 16, 'prize': 17, 'this': 18, 'is': 19, 'short': 20, 'sentence': 21}

Sequence 1 in x
  Input:  The quick brown fox jumps over the lazy dog .
  Output: [1, 2, 4, 5, 6, 7, 1, 8, 9]
Sequence 2 in x
  Input:  By Jove , my quick study of lexicography won a prize .
  Output: [10, 11, 12, 2, 13, 14, 15, 16, 3, 17]
Sequence 3 in x
  Input:  This is a short sentence .
  Output: [18, 19, 3, 20, 21]
```

For padding we will use the pad_sequence of keras. This lets us make the lengths of english sentences and french sentences equal.



```
Sequence 1 in x
  Input:  [1 2 4 5 6 7 1 8 9]
  Output: [1 2 4 5 6 7 1 8 9 0]
Sequence 2 in x
  Input:  [10 11 12  2 13 14 15 16  3 17]
  Output: [10 11 12  2 13 14 15 16  3 17]
Sequence 3 in x
  Input:  [18 19  3 20 21]
  Output: [18 19  3 20 21  0  0  0  0  0]
```

The final assessment of the Vocab:



```
Data Preprocessed
Max English sentence length: 15
Max French sentence length: 21
English vocabulary size: 12780
French vocabulary size: 25465
```

5.2.2 Training the Models

Simple RNN

This is the most basic type of RNN.

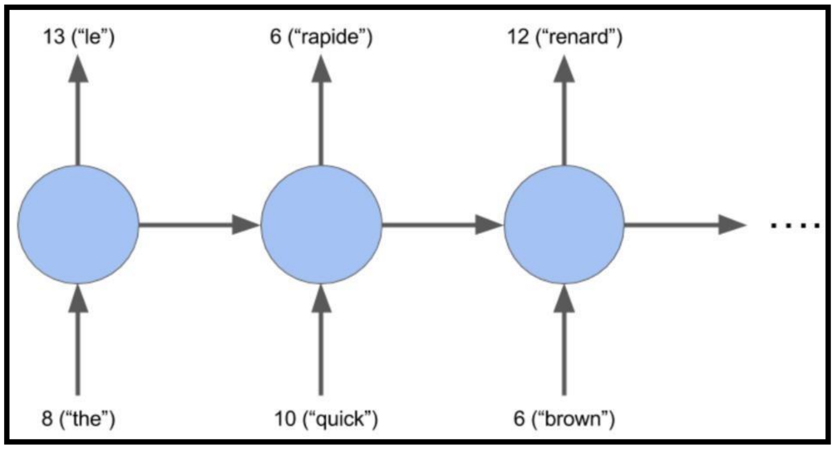
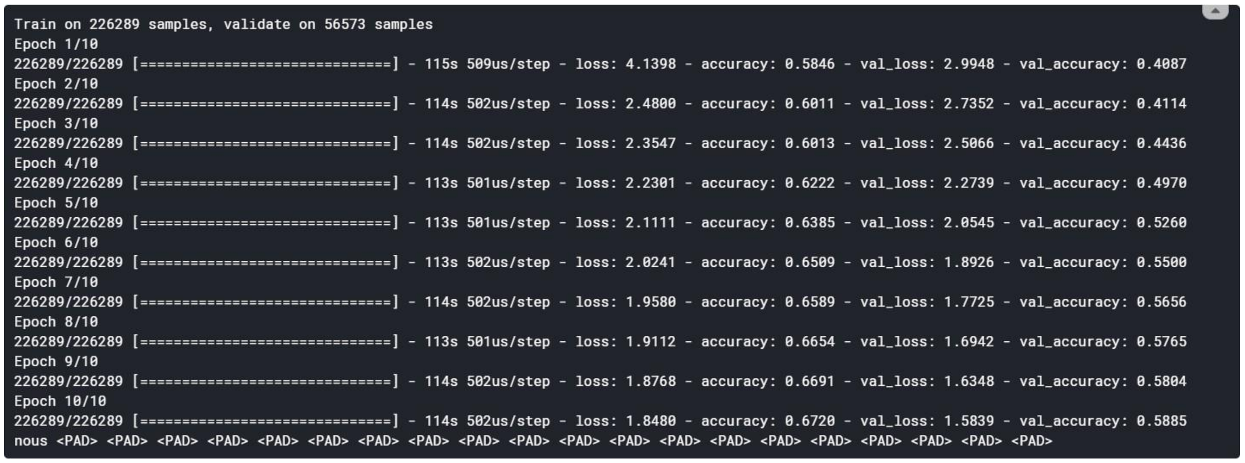


Fig. Simple Recurrent Neural Network

For this model we have observed the time and accuracy of translation by using 10 Epochs.



```
Train on 226289 samples, validate on 56573 samples
Epoch 1/10
226289/226289 [==============================] - 115s 509us/step - loss: 4.1398 - accuracy: 0.5846 - val_loss: 2.9948 - val_accuracy: 0.4087
Epoch 2/10
226289/226289 [==============================] - 114s 502us/step - loss: 2.4800 - accuracy: 0.6011 - val_loss: 2.7352 - val_accuracy: 0.4114
Epoch 3/10
226289/226289 [==============================] - 114s 502us/step - loss: 2.3547 - accuracy: 0.6013 - val_loss: 2.5066 - val_accuracy: 0.4436
Epoch 4/10
226289/226289 [==============================] - 113s 501us/step - loss: 2.2301 - accuracy: 0.6222 - val_loss: 2.2739 - val_accuracy: 0.4970
Epoch 5/10
226289/226289 [==============================] - 113s 501us/step - loss: 2.1111 - accuracy: 0.6385 - val_loss: 2.0545 - val_accuracy: 0.5260
Epoch 6/10
226289/226289 [==============================] - 113s 502us/step - loss: 2.0241 - accuracy: 0.6509 - val_loss: 1.8926 - val_accuracy: 0.5500
Epoch 7/10
226289/226289 [==============================] - 114s 502us/step - loss: 1.9580 - accuracy: 0.6589 - val_loss: 1.7725 - val_accuracy: 0.5656
Epoch 8/10
226289/226289 [==============================] - 113s 501us/step - loss: 1.9112 - accuracy: 0.6654 - val_loss: 1.6942 - val_accuracy: 0.5765
Epoch 9/10
226289/226289 [==============================] - 114s 502us/step - loss: 1.8768 - accuracy: 0.6691 - val_loss: 1.6348 - val_accuracy: 0.5804
Epoch 10/10
226289/226289 [==============================] - 114s 502us/step - loss: 1.8480 - accuracy: 0.6720 - val_loss: 1.5839 - val_accuracy: 0.5885
nous <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Therefore, we observe that simple RNN has an accuracy of 58.85%.

RNN with Embedding

This model involves a technique of creating a vector representation of unique tokens that are similar or have similar meaning in a m-dimensional space.

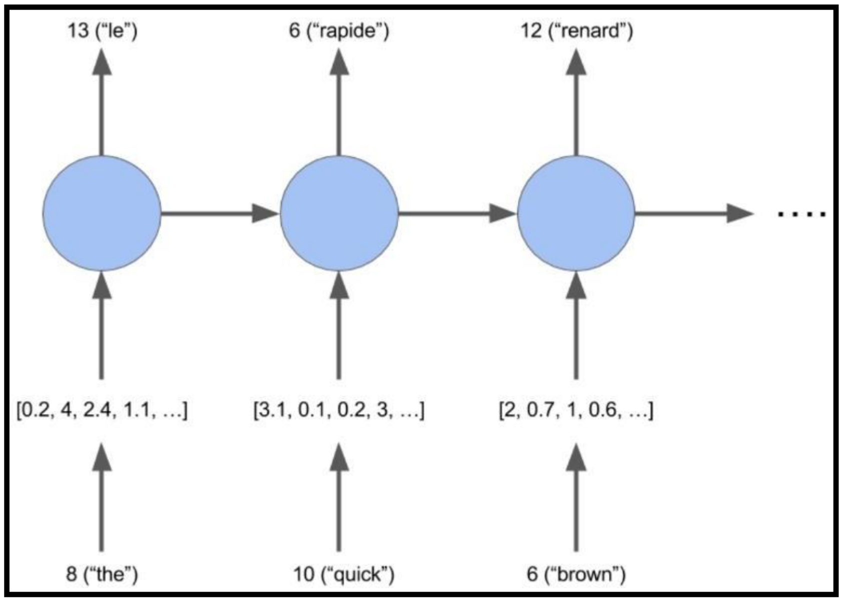
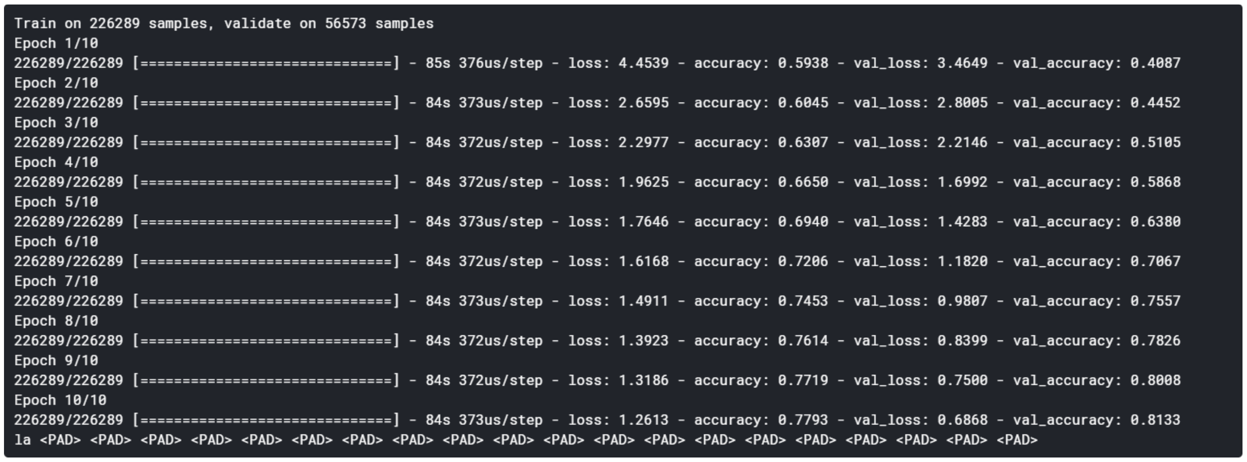


Fig. RNN with Embedding

For this model we have observed the time and accuracy of translation by using 10 Epochs.



```
Train on 226289 samples, validate on 56573 samples
Epoch 1/10
226289/226289 [==============================] - 85s 376us/step - loss: 4.4539 - accuracy: 0.5938 - val_loss: 3.4649 - val_accuracy: 0.4087
Epoch 2/10
226289/226289 [==============================] - 84s 373us/step - loss: 2.6595 - accuracy: 0.6045 - val_loss: 2.8005 - val_accuracy: 0.4452
Epoch 3/10
226289/226289 [==============================] - 84s 372us/step - loss: 2.2977 - accuracy: 0.6307 - val_loss: 2.2146 - val_accuracy: 0.5105
Epoch 4/10
226289/226289 [==============================] - 84s 372us/step - loss: 1.9625 - accuracy: 0.6650 - val_loss: 1.6992 - val_accuracy: 0.5868
Epoch 5/10
226289/226289 [==============================] - 84s 373us/step - loss: 1.7646 - accuracy: 0.6940 - val_loss: 1.4283 - val_accuracy: 0.6380
Epoch 6/10
226289/226289 [==============================] - 84s 372us/step - loss: 1.6168 - accuracy: 0.7206 - val_loss: 1.1820 - val_accuracy: 0.7067
Epoch 7/10
226289/226289 [==============================] - 84s 373us/step - loss: 1.4911 - accuracy: 0.7453 - val_loss: 0.9807 - val_accuracy: 0.7557
Epoch 8/10
226289/226289 [==============================] - 84s 372us/step - loss: 1.3923 - accuracy: 0.7614 - val_loss: 0.8399 - val_accuracy: 0.7826
Epoch 9/10
226289/226289 [==============================] - 84s 372us/step - loss: 1.3186 - accuracy: 0.7719 - val_loss: 0.7500 - val_accuracy: 0.8008
Epoch 10/10
226289/226289 [==============================] - 84s 373us/step - loss: 1.2613 - accuracy: 0.7793 - val_loss: 0.6868 - val_accuracy: 0.8133
la <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Therefore, we observe that simple RNN has an accuracy of 81.33%.

Bidirectional RNN

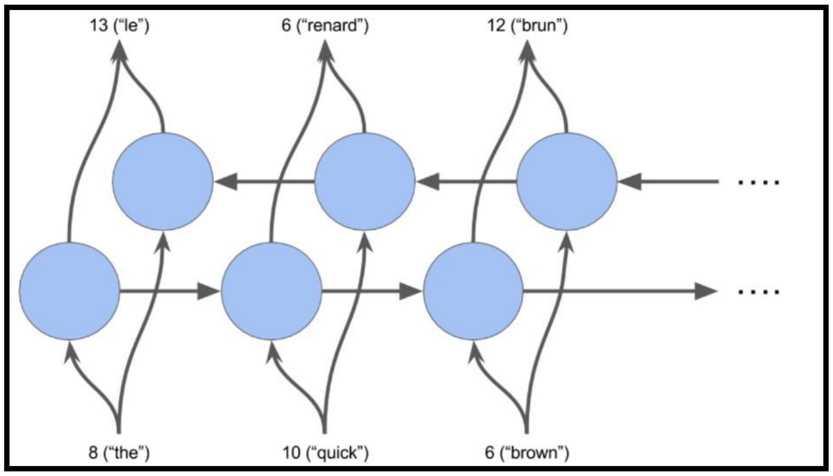
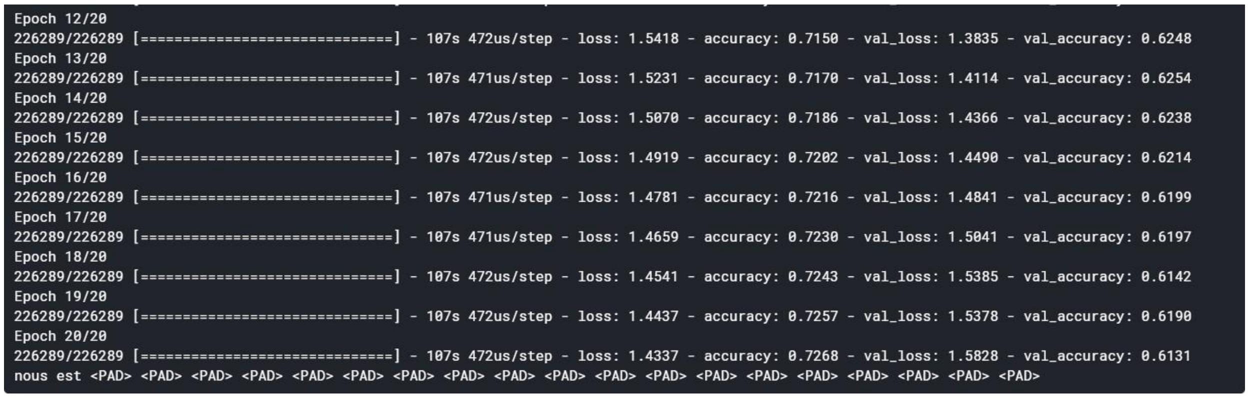


Fig. Bidirectional RNN

For this model we have observed the time and accuracy of translation by using 20 Epochs.

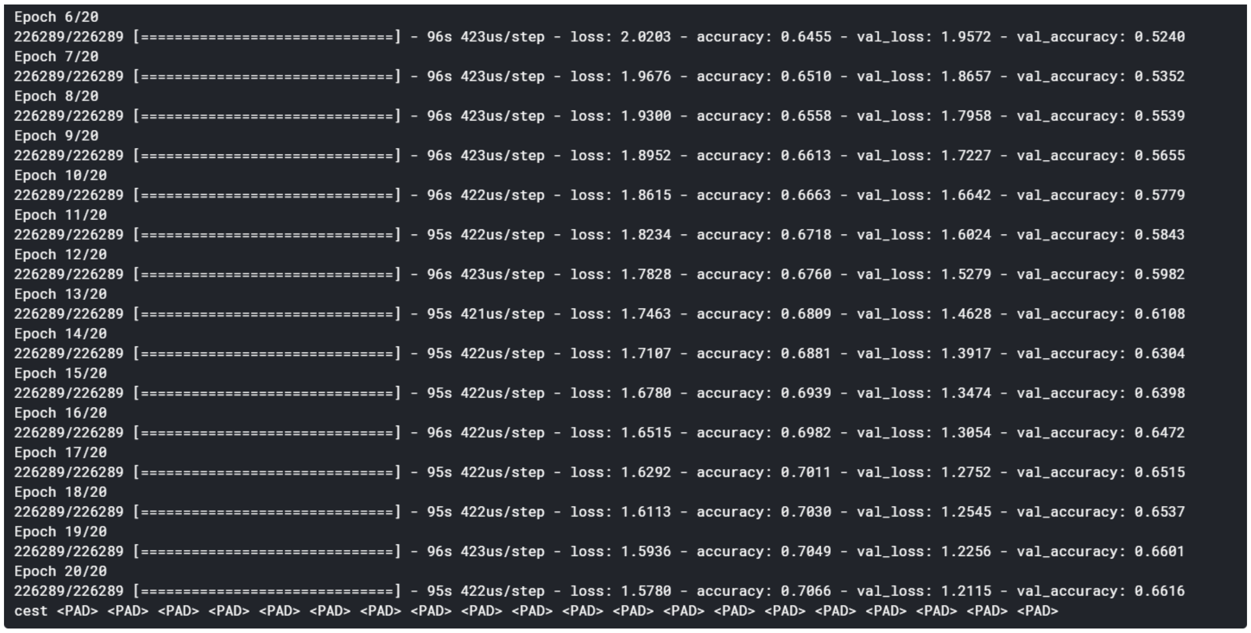


```
Epoch 12/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.5418 - accuracy: 0.7150 - val_loss: 1.3835 - val_accuracy: 0.6248
Epoch 13/20
226289/226289 [==============================] - 107s 471us/step - loss: 1.5231 - accuracy: 0.7170 - val_loss: 1.4114 - val_accuracy: 0.6254
Epoch 14/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.5070 - accuracy: 0.7186 - val_loss: 1.4366 - val_accuracy: 0.6238
Epoch 15/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.4919 - accuracy: 0.7202 - val_loss: 1.4490 - val_accuracy: 0.6214
Epoch 16/20
226289/226289 [==============================] - 107s 471us/step - loss: 1.4781 - accuracy: 0.7216 - val_loss: 1.4841 - val_accuracy: 0.6199
Epoch 17/20
226289/226289 [==============================] - 107s 471us/step - loss: 1.4659 - accuracy: 0.7230 - val_loss: 1.5041 - val_accuracy: 0.6197
Epoch 18/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.4541 - accuracy: 0.7243 - val_loss: 1.5385 - val_accuracy: 0.6142
Epoch 19/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.4437 - accuracy: 0.7257 - val_loss: 1.5378 - val_accuracy: 0.6190
Epoch 20/20
226289/226289 [==============================] - 107s 472us/step - loss: 1.4337 - accuracy: 0.7268 - val_loss: 1.5828 - val_accuracy: 0.6131
nous est <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Therefore, we observe that simple RNN has an accuracy of 61.31%. We can also observe that the accuracy decreases after the 9th Epoch. This is due to overfitting so there is no point increasing the number of Epoch to increase the accuracy of the model.

Encoder - Decoder RNN

For this model we have observed the time and accuracy of translation by using 20 Epochs.

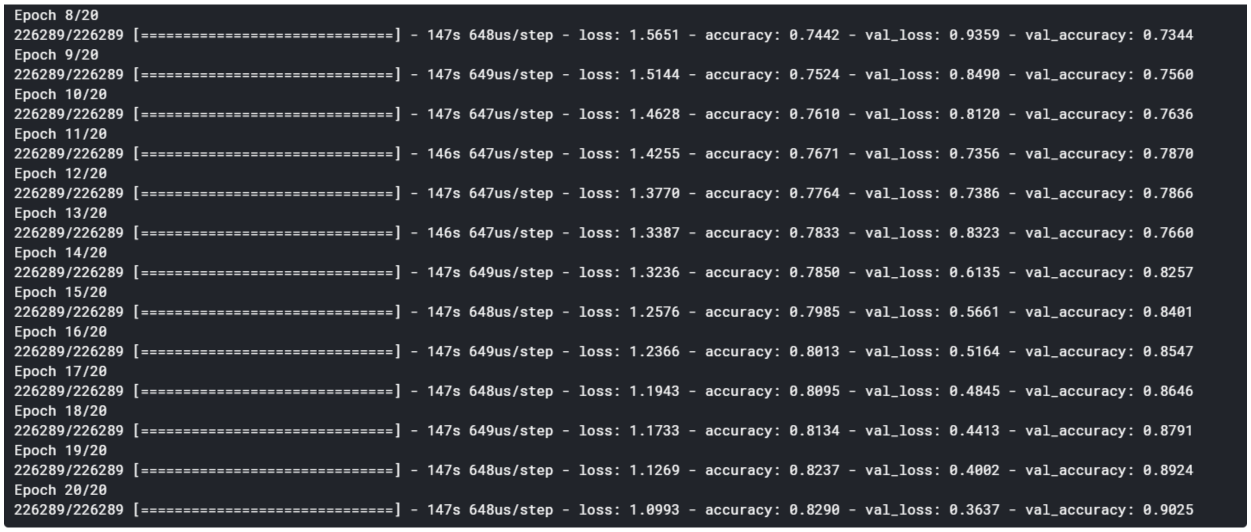


```
Epoch 6/20
226289/226289 [==============================] - 96s 423us/step - loss: 2.0203 - accuracy: 0.6455 - val_loss: 1.9572 - val_accuracy: 0.5240
Epoch 7/20
226289/226289 [==============================] - 96s 423us/step - loss: 1.9676 - accuracy: 0.6510 - val_loss: 1.8657 - val_accuracy: 0.5352
Epoch 8/20
226289/226289 [==============================] - 96s 423us/step - loss: 1.9300 - accuracy: 0.6558 - val_loss: 1.7958 - val_accuracy: 0.5539
Epoch 9/20
226289/226289 [==============================] - 96s 423us/step - loss: 1.8952 - accuracy: 0.6613 - val_loss: 1.7227 - val_accuracy: 0.5655
Epoch 10/20
226289/226289 [==============================] - 96s 422us/step - loss: 1.8615 - accuracy: 0.6663 - val_loss: 1.6642 - val_accuracy: 0.5779
Epoch 11/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.8234 - accuracy: 0.6718 - val_loss: 1.6024 - val_accuracy: 0.5843
Epoch 12/20
226289/226289 [==============================] - 96s 423us/step - loss: 1.7828 - accuracy: 0.6760 - val_loss: 1.5279 - val_accuracy: 0.5982
Epoch 13/20
226289/226289 [==============================] - 95s 421us/step - loss: 1.7463 - accuracy: 0.6809 - val_loss: 1.4628 - val_accuracy: 0.6108
Epoch 14/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.7107 - accuracy: 0.6881 - val_loss: 1.3917 - val_accuracy: 0.6304
Epoch 15/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.6780 - accuracy: 0.6939 - val_loss: 1.3474 - val_accuracy: 0.6398
Epoch 16/20
226289/226289 [==============================] - 96s 422us/step - loss: 1.6515 - accuracy: 0.6982 - val_loss: 1.3054 - val_accuracy: 0.6472
Epoch 17/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.6292 - accuracy: 0.7011 - val_loss: 1.2752 - val_accuracy: 0.6515
Epoch 18/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.6113 - accuracy: 0.7030 - val_loss: 1.2545 - val_accuracy: 0.6537
Epoch 19/20
226289/226289 [==============================] - 96s 423us/step - loss: 1.5936 - accuracy: 0.7049 - val_loss: 1.2256 - val_accuracy: 0.6601
Epoch 20/20
226289/226289 [==============================] - 95s 422us/step - loss: 1.5780 - accuracy: 0.7066 - val_loss: 1.2115 - val_accuracy: 0.6616
cest <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Therefore, we observe that simple RNN has an accuracy of 66.16%.

Custom Model (Embedded + Bidirectional)

For this model we have observed the time and accuracy of translation by using 20 Epochs.



```
Epoch 8/20
226289/226289 [==============================] - 147s 648us/step - loss: 1.5651 - accuracy: 0.7442 - val_loss: 0.9359 - val_accuracy: 0.7344
Epoch 9/20
226289/226289 [==============================] - 147s 649us/step - loss: 1.5144 - accuracy: 0.7524 - val_loss: 0.8490 - val_accuracy: 0.7560
Epoch 10/20
226289/226289 [==============================] - 147s 647us/step - loss: 1.4628 - accuracy: 0.7610 - val_loss: 0.8120 - val_accuracy: 0.7636
Epoch 11/20
226289/226289 [==============================] - 146s 647us/step - loss: 1.4255 - accuracy: 0.7671 - val_loss: 0.7356 - val_accuracy: 0.7870
Epoch 12/20
226289/226289 [==============================] - 147s 647us/step - loss: 1.3770 - accuracy: 0.7764 - val_loss: 0.7386 - val_accuracy: 0.7866
Epoch 13/20
226289/226289 [==============================] - 146s 647us/step - loss: 1.3387 - accuracy: 0.7833 - val_loss: 0.8323 - val_accuracy: 0.7660
Epoch 14/20
226289/226289 [==============================] - 147s 649us/step - loss: 1.3236 - accuracy: 0.7850 - val_loss: 0.6135 - val_accuracy: 0.8257
Epoch 15/20
226289/226289 [==============================] - 147s 648us/step - loss: 1.2576 - accuracy: 0.7985 - val_loss: 0.5661 - val_accuracy: 0.8401
Epoch 16/20
226289/226289 [==============================] - 147s 649us/step - loss: 1.2366 - accuracy: 0.8013 - val_loss: 0.5164 - val_accuracy: 0.8547
Epoch 17/20
226289/226289 [==============================] - 147s 648us/step - loss: 1.1943 - accuracy: 0.8095 - val_loss: 0.4845 - val_accuracy: 0.8646
Epoch 18/20
226289/226289 [==============================] - 147s 649us/step - loss: 1.1733 - accuracy: 0.8134 - val_loss: 0.4413 - val_accuracy: 0.8791
Epoch 19/20
226289/226289 [==============================] - 147s 648us/step - loss: 1.1269 - accuracy: 0.8237 - val_loss: 0.4002 - val_accuracy: 0.8924
Epoch 20/20
226289/226289 [==============================] - 147s 648us/step - loss: 1.0993 - accuracy: 0.8290 - val_loss: 0.3637 - val_accuracy: 0.9025
```

Therefore, we observe that simple RNN has an accuracy of 90.25%. Thus, we observe that our custom model is the best among the used models. We will save this model and use it for the translation.

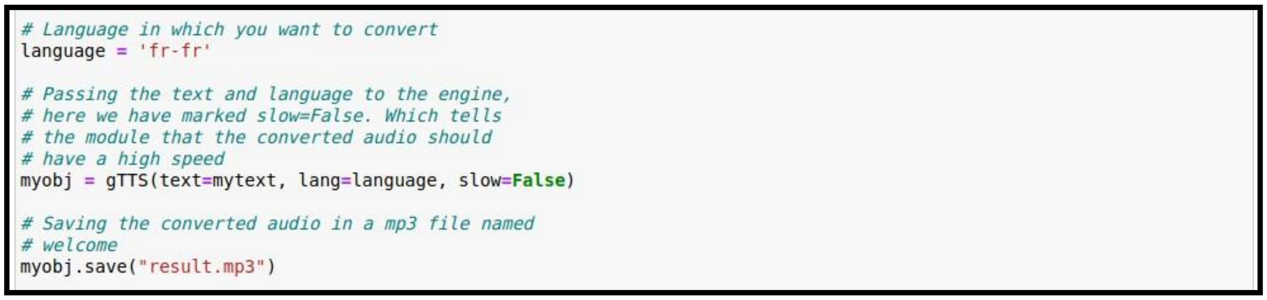
5.2.3 Final Accuracies of the Models used

Model	Epochs	Accuracy (in percentage)
Simple RNN	10	58.85
RNN with Embedding	10	81.33
Bidirectional RNN	20	61.31
Encoder - Decoder RNN	20	66.16
Custom Model (Embedded + Bidirectional)	20	90.25

NOTE: We will save the custom model for further use.

5.3 Converting text to speech

We preprocess the dataset and load the saved model. We get the translation by using model.predict(). The translation is stored in a variable for further use. For this third module we will use gTTs (google Text-to-Speech) to convert this text into audio file, save the audio file and then play it.



```
# Language in which you want to convert
language = 'fr-fr'

# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=mytext, lang=language, slow=False)

# Saving the converted audio in a mp3 file named
# welcome
myobj.save("result.mp3")
```

RESULTS AND DISCUSSION

The system has been built with several RNN models, namely the Basic Recurrent Neural Network model, Recurrent Neural Network with Embedding, Bidirectional Recurrent Neural Network, and Recurrent Neural Network with embedded encoder and decoder.

For visualizing the tabulated data, the python library matplotlib has been used for multiple line plot. Matplotlib is an open source python library which is widely used to deal with visualization of data in order to get better insight and analysis of the data. The system has been built over Google Colab which provides cloud services to execute codes, along with hardware supports with GPU.

Table-I: Accuracy percentage vs epoch for system models

Epoch Number	**Model Accuracy in %**			
	Basic RNN	***RNN with Encoding***	***Bidirectional RNN***	***RNN Encoder and Decoder***
1	41.99	40.02	49.94	44.00
2	47.07	44.49	59.04	49.78
3	52.78	55.02	61.50	51.06
4	56.98	64.14	63.10	53.01
5	58.42	71.62	64.47	55.84
6	58.82	76.32	65.38	57.24
7	59.27	78.79	66.74	58.29
8	60.12	80.58	67.23	59.24
9	61.33	81.92	67.64	60.11
10	62.07	83.18	68.08	60.77

Table-I shows tabular comparison of accuracy percentages achieved for every round of the epoch for all of the models that are used in the system. The visualization of the data obtained through the plot shown in Fig. 2, where accuracy versus the iteration number of epochs is plotted in order to figure how the accuracy is increasing with every iteration for the four models.

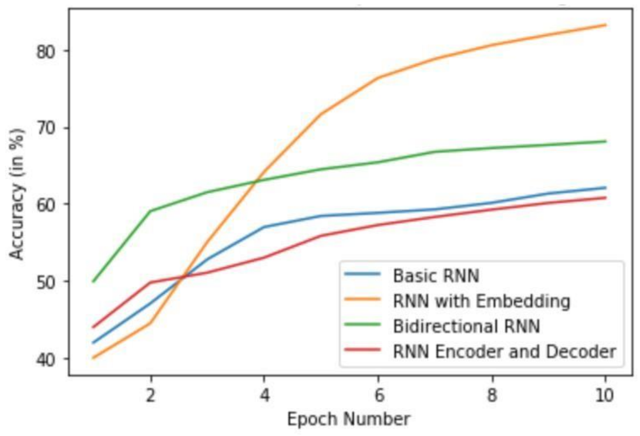


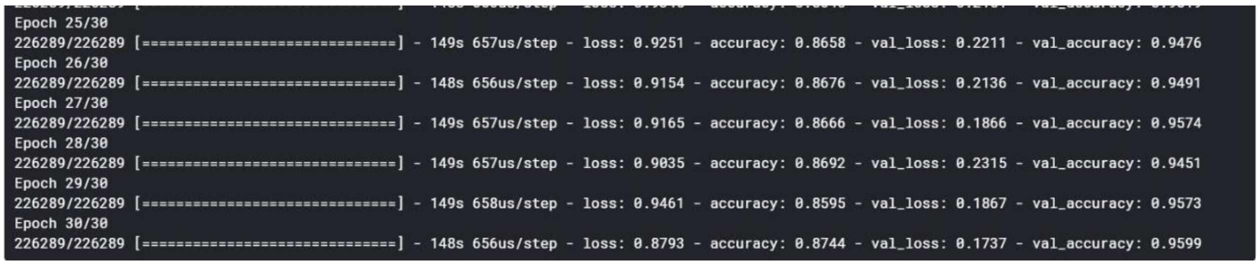
Figure 2. Plot of accuracy vs. epoch number for models

From the plot it can be clearly seen that the Recurrent Neural Network with encoding is having the highest accuracy with increase in iterations, bidirectional Recurrent Neural Network gives good amount of accuracy with time, and is more or less constant in nature, the same way, basic Recurrent Neural Network and Recurrent Neural Network with encoder and decoders also give kind of constant accuracy, but comparatively it is lower than the other two models. The final system combines the top two models giving the highest accuracy, in order to boost up the accuracy.

LIST OF CHALLENGES FACED AND SNAPSHOTS OF ERRORS

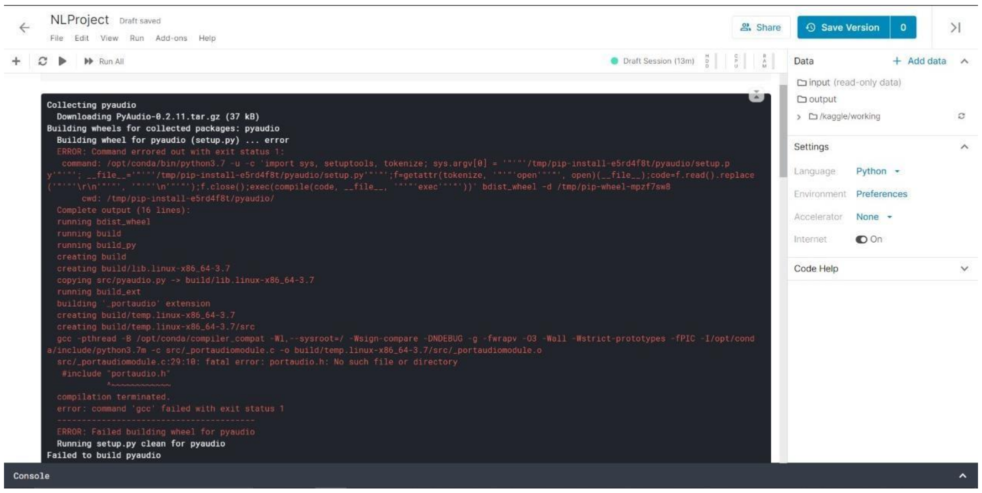
1. When we increased the number of epoch to greater than 30 though the validation accuracy increased when tested on new data it was not able to perform adequately despite the high 95% accuracy. After changing some parameters and reducing epochs we came to the conclusion that the above difficulty was due to the model overfitting on the data. To resolve this we used a maximum of 20 epochs and tried different model parameters

2. Training of the model using CPU is slow thus we used the free GPU provided in Kaggle for the training. The Kaggle GPU was able to speed up the process, hence all training was there on Kaggle kernel



```
Epoch 25/30
226289/226289 [==============================] - 149s 657us/step - loss: 0.9251 - accuracy: 0.8658 - val_loss: 0.2211 - val_accuracy: 0.9476
Epoch 26/30
226289/226289 [==============================] - 148s 656us/step - loss: 0.9154 - accuracy: 0.8676 - val_loss: 0.2136 - val_accuracy: 0.9491
Epoch 27/30
226289/226289 [==============================] - 149s 657us/step - loss: 0.9165 - accuracy: 0.8666 - val_loss: 0.1866 - val_accuracy: 0.9574
Epoch 28/30
226289/226289 [==============================] - 149s 657us/step - loss: 0.9035 - accuracy: 0.8692 - val_loss: 0.2315 - val_accuracy: 0.9451
Epoch 29/30
226289/226289 [==============================] - 149s 658us/step - loss: 0.9461 - accuracy: 0.8595 - val_loss: 0.1867 - val_accuracy: 0.9573
Epoch 30/30
226289/226289 [==============================] - 148s 656us/step - loss: 0.8793 - accuracy: 0.8744 - val_loss: 0.1737 - val_accuracy: 0.9599
```

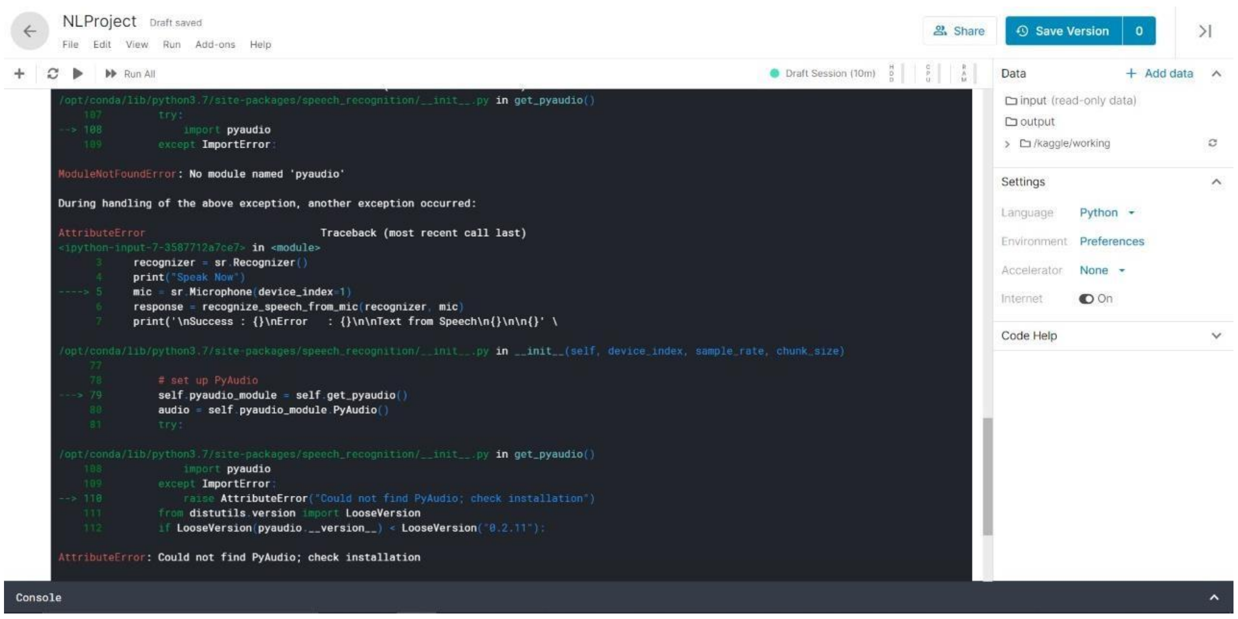
Kaggle does not support the use of Google Speech Recognition API as it requires Pyaudio, which is why the trained model is saved and tested using Jupyter Notebook.



Language: Python
Environment: Preferences
Accelerator: None
Internet: On

```
Collecting pyaudio
  Downloading PyAudio-0.2.11.tar.gz (37 kB)
Building wheels for collected packages: pyaudio
  Building wheel for pyaudio (setup.py) ... error
  ERROR: Command errored out with exit status 1:
   command: /opt/conda/bin/python3.7 -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-e5rd4f8t/pyaudio/setup.p
y'"'"'; __file__='"'"'/tmp/pip-install-e5rd4f8t/pyaudio/setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace
('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' bdist_wheel -d /tmp/pip-wheel-mpzf7sw8
       cwd: /tmp/pip-install-e5rd4f8t/pyaudio/
  Complete output (16 lines):
  running bdist_wheel
  running build
  running build_py
  creating build
  creating build/lib.linux-x86_64-3.7
  copying src/pyaudio.py -> build/lib.linux-x86_64-3.7
  running build_ext
  building '_portaudio' extension
  creating build/temp.linux-x86_64-3.7
  creating build/temp.linux-x86_64-3.7/src
  gcc -pthread -B /opt/conda/compiler_compat -Wl,--sysroot=/ -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/opt/cond
a/include/python3.7m -c src/_portaudiomodule.c -o build/temp.linux-x86_64-3.7/src/_portaudiomodule.o
  src/_portaudiomodule.c:29:10: fatal error: portaudio.h: No such file or directory
   #include "portaudio.h"
            ^~~~~~~~~~~~~
  compilation terminated.
  error: command 'gcc' failed with exit status 1
  ----------------------------------------
  ERROR: Failed building wheel for pyaudio
  Running setup.py clean for pyaudio
Failed to build pyaudio
```

3. To import the speech_recognition package of SpeechRecognition Library we had to download pyaudio for it to run even though we are not explicitly importing it.

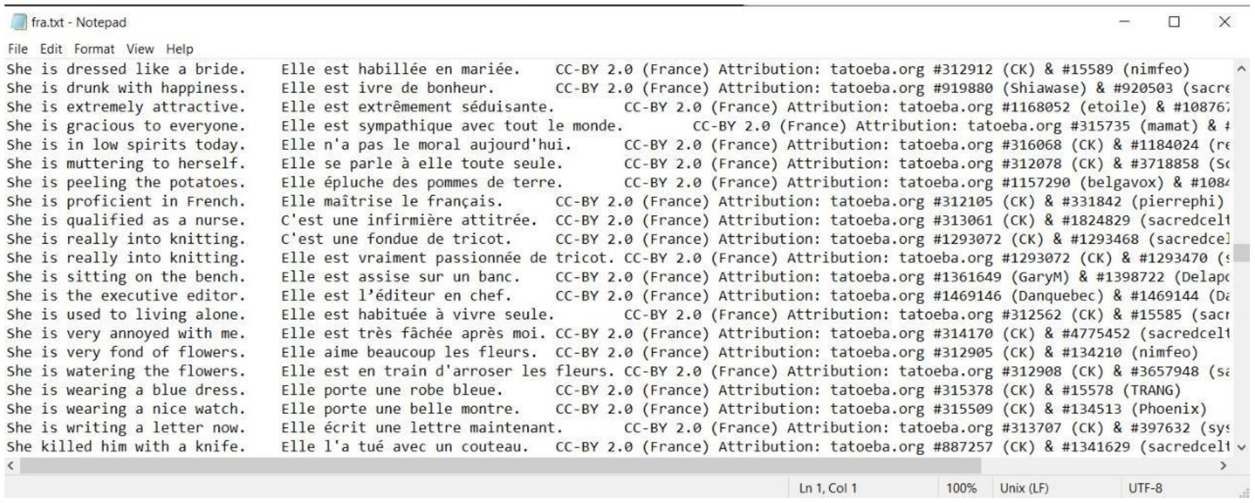


Settings

Language: Python
Environment: Preferences
Accelerator: None
Internet: On

```
/opt/conda/lib/python3.7/site-packages/speech_recognition/__init__.py in get_pyaudio()
    107         try:
--> 108             import pyaudio
    109         except ImportError:

ModuleNotFoundError: No module named 'pyaudio'

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
<ipython-input-7-3587712a7ce7> in <module>
      3     recognizer = sr.Recognizer()
      4     print("Speak Now")
----> 5     mic = sr.Microphone(device_index=1)
      6     response = recognize_speech_from_mic(recognizer, mic)
      7     print('\nSuccess : {}\nError   : {}\n\nText from Speech\n{}\n\n{}' \

/opt/conda/lib/python3.7/site-packages/speech_recognition/__init__.py in __init__(self, device_index, sample_rate, chunk_size)
     77 
     78         # set up PyAudio
---> 79         self.pyaudio_module = self.get_pyaudio()
     80         audio = self.pyaudio_module.PyAudio()
     81         try:

/opt/conda/lib/python3.7/site-packages/speech_recognition/__init__.py in get_pyaudio()
    108             import pyaudio
    109         except ImportError:
--> 110             raise AttributeError("Could not find PyAudio; check installation")
    111         from distutils.version import LooseVersion
    112         if LooseVersion(pyaudio.__version__) < LooseVersion("0.2.11"):

AttributeError: Could not find PyAudio; check installation
```

4. Making the compiled dataset was a challenge as our primary source (http://www.manythings.org/anki/) had English and French sentence in different files. We needed them in same files along with a mapping so that we can train it in the model. The above screenshot shows the final custom dataset which we finally were able to create.



```
She is dressed like a bride.	Elle est habillée en mariée.	CC-BY 2.0 (France) Attribution: tatoeba.org #312912 (CK) & #15589 (nimfeo)
She is drunk with happiness.	Elle est ivre de bonheur.	CC-BY 2.0 (France) Attribution: tatoeba.org #919880 (Shiawase) & #920503 (sacre
She is extremely attractive.	Elle est extrêmement séduisante.	CC-BY 2.0 (France) Attribution: tatoeba.org #1168052 (etoile) & #10876
She is gracious to everyone.	Elle est sympathique avec tout le monde.	CC-BY 2.0 (France) Attribution: tatoeba.org #315735 (mamat) & #
She is in low spirits today.	Elle n'a pas le moral aujourd'hui.	CC-BY 2.0 (France) Attribution: tatoeba.org #316068 (CK) & #1184024 (re
She is muttering to herself.	Elle se parle à elle toute seule.	CC-BY 2.0 (France) Attribution: tatoeba.org #312078 (CK) & #3718858 (S
She is peeling the potatoes.	Elle épluche des pommes de terre.	CC-BY 2.0 (France) Attribution: tatoeba.org #1157290 (belgavox) & #108
She is proficient in French.	Elle maîtrise le français.	CC-BY 2.0 (France) Attribution: tatoeba.org #312105 (CK) & #331842 (pierrephi)
She is qualified as a nurse.	C'est une infirmière attitrée.	CC-BY 2.0 (France) Attribution: tatoeba.org #313061 (CK) & #1824829 (sacredcel
She is really into knitting.	C'est une fondue de tricot.	CC-BY 2.0 (France) Attribution: tatoeba.org #1293072 (CK) & #1293468 (sacredce
She is really into knitting.	Elle est vraiment passionnée de tricot.	CC-BY 2.0 (France) Attribution: tatoeba.org #1293072 (CK) & #1293470 (
She is sitting on the bench.	Elle est assise sur un banc.	CC-BY 2.0 (France) Attribution: tatoeba.org #1361649 (GaryM) & #1398722 (Delap
She is the executive editor.	Elle est l'éditeur en chef.	CC-BY 2.0 (France) Attribution: tatoeba.org #1469146 (Danquebec) & #1469144 (D
She is used to living alone.	Elle est habituée à vivre seule.	CC-BY 2.0 (France) Attribution: tatoeba.org #312562 (CK) & #15585 (sac
She is very annoyed with me.	Elle est très fâchée après moi.	CC-BY 2.0 (France) Attribution: tatoeba.org #314170 (CK) & #4775452 (sacredcel
She is very fond of flowers.	Elle aime beaucoup les fleurs.	CC-BY 2.0 (France) Attribution: tatoeba.org #312905 (CK) & #134210 (nimfeo)
She is watering the flowers.	Elle est en train d'arroser les fleurs.	CC-BY 2.0 (France) Attribution: tatoeba.org #312908 (CK) & #3657948 (s
She is wearing a blue dress.	Elle porte une robe bleue.	CC-BY 2.0 (France) Attribution: tatoeba.org #315378 (CK) & #15578 (TRANG)
She is wearing a nice watch.	Elle porte une belle montre.	CC-BY 2.0 (France) Attribution: tatoeba.org #315509 (CK) & #134513 (Phoenix)
She is writing a letter now.	Elle écrit une lettre maintenant.	CC-BY 2.0 (France) Attribution: tatoeba.org #313707 (CK) & #397632 (sy
She killed him with a knife.	Elle l'a tué avec un couteau.	CC-BY 2.0 (France) Attribution: tatoeba.org #887257 (CK) & #1341629 (sacredcel
```

CONCLUSION

We expect three results in our project based on the three different modules. From the first module we expect the code to analyse and process the speech of the humans by removing the ambient noises and convert it into text format. Once this is done the transformed text is transferred to the second module. The second module is supposed to use the model with best accuracy to translate the text received to the required language. The translated text is then forwarded to the third module where it is supposed to synthesize the speech i.e. read it on behalf of you.
Nowadays people love to travel and the major issue faced by them is the language. Many companies hire special human translators to help them understand their foreign clients and complete the deals. Our project aims at helping such people and businesses without any major hardware requirements.

REFERENCES

[1] Review of Speech-to-Text Recognition Technology for Enhancing Learning - *Rustam Shadiev1 , Wu-Yuin Hwang2 , Nian-Shing Chen3 and Yueh-Min Huang(2014)*
[2] Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition -*Hagen Soltau, Hank Liao, Hasim Sak(2016)*
[3] Achieving Human Parity in Conversation - *W. Xiong, J. Droppo, X. Huang, F. Seide*
[4] Neural Responding Machine for Short-Text Conversation - *Lifeng Shang Zhengdong Lu Hang Li*
[5] Evaluating GRU and LSTM with Regularization on Translating Different Language Pairs - *Shi Fan, Yili Yu*
[6] *Battenberg, E., Chen, J., Child, R., Coates, A., Li, Y. G. Y., Liu, H., ... Zhu, Z. (2017)*. Exploring neural transducers for end-to-end speech recognition. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). doi: 10.1109/asru.2017.8268937
[7] Unsupervised Neural Machine Translation.Mikel Artetxe, Gorka Labaka, Eneko Agirre, Kyunghyun Cho. (2018, February). Published as a conference paper at ICLR 2018.
[8] A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation.Surafel M. Lakew, Mauro Cettolo, Marcello Federico. (2018, June). The 27th International Conference on Computational Linguistics (COLING 2018). arXiv:1806.06957 [cs.CL]
[9] AI Speech Recognition - *M. Seltzer, A. Stolcke, D. Yu and G. Zweig*
[10] *Kitano, H. (1994).* Current Research Toward Speech-to-Speech Translation. Speech-to-Speech Translation, 13–27. doi: 10.1007/978-1-4615-2732-9_2

GitHub Repository Link

https://github.com/Samridhi98/NLP_JComponent.git