INDIE-LENS

MINOR PROJECT

Submitted By -

Kanika Gupta (14103271)
Stuti (14103056)
B10

Under the supervision of -

Prof. Vimal Kumar

Problem Statement

Travelled to India and having difficulty in comprehending what the pictures, posters or signboards around have written on them? To solve this is the primary purpose of our application. The user has to click a picture with and leave the rest on the application. He will instantly get an English translation of the text written in Hindi. Apart from this, this application is an overall package for a person traveling to India with other features like:

- He can also input text in English.
- He can search the nearby places using his location.
- Can make his own travel kit list to keep a track.

Introduction

The main reason and objective of this app is to make travelling easier and not letting language be a barrier. The whole application will be using Artificial Intelligence as a backdrop.
The app will be using Optical Character Recognition (OCR) to detect the text in the picture clicked which will then be translated using Artificial Intelligence. Other features of the project will require only android studio as a platform.
The artificial intelligence algorithm used in our project is LSTM Neural Network.

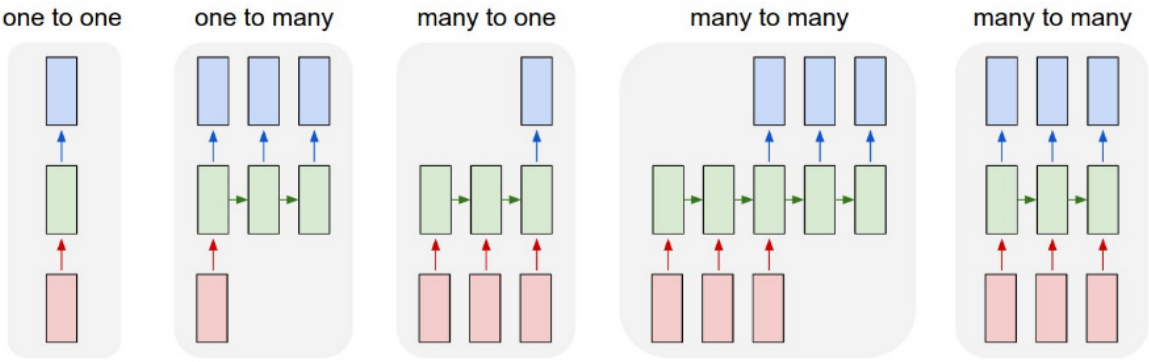
Software Requirements

1. Android Studio
2. Android Emulator 26.0.0
3. Google Play Services
4. Google Places API
5. Mobile Vision Text API
6. SQLite Database
7. Java JDK 8.0
8. Tensorflow 1.0.0
9. Keras 1.1.0
10. Numpy
11. Scipy
12. Gensim
13. Python 3.6

Background Study and Findings

Artificial Intelligence

On understanding the need of our project, we realised what we were required to apply was Artificial Intelligence in order to train my application to translate a set of characters in Hindi from English. For the same we first need to encode the English input in vectors which will then be mapped to the output it should and there on decoded from a set of vectors to a set of characters in Hindi.

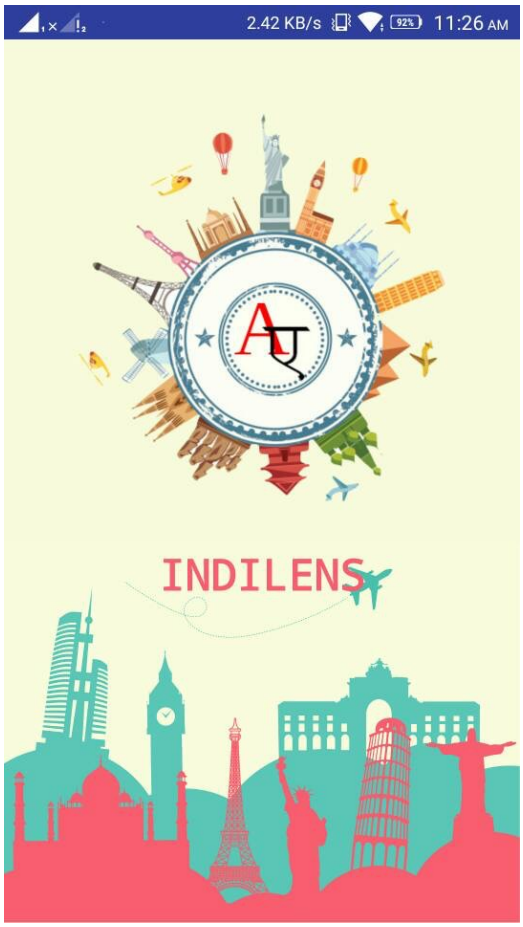


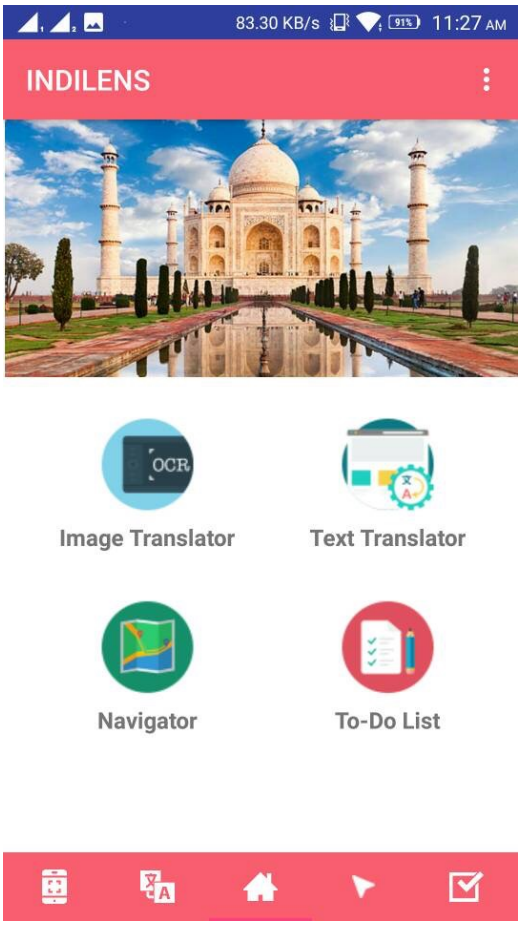
OCR

To translate text written on signs, billboards and posters, we needed a way to extract text from an image and convert it into characters. OCR - Optical Character Recognition/ Reader helps us achieve the same, by identifying the areas where text is present in the image and then converting the text in the form of text blocks which can be further used for translation.

System Design in Detail

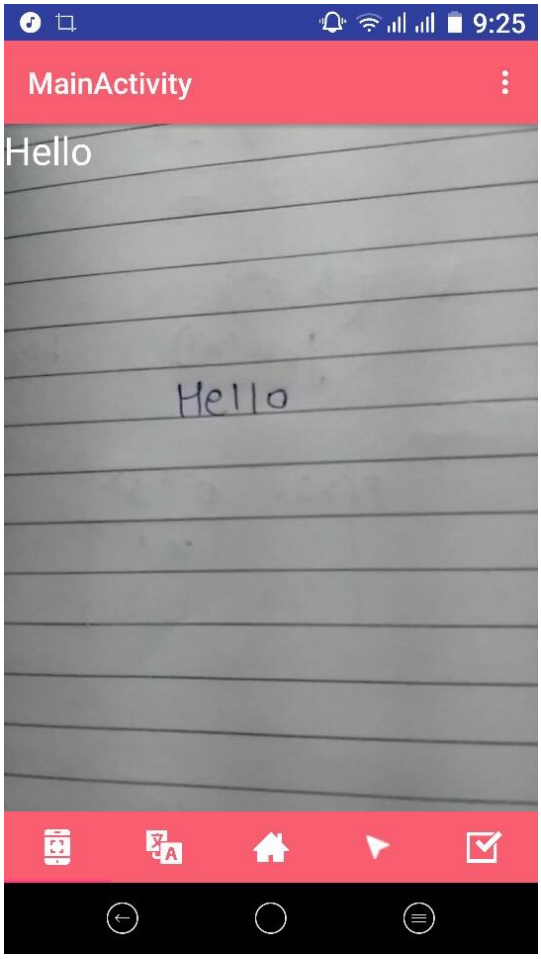
The app opens with a splash screen displaying the logo of IndieLens which then switches to the home screen which has the menu - a grid view of all the main functionalities of the application. The whole application is a tabbed activity, the user can switch between widgets by sliding or using the navigation bar at the bottom for easy access to the various features.

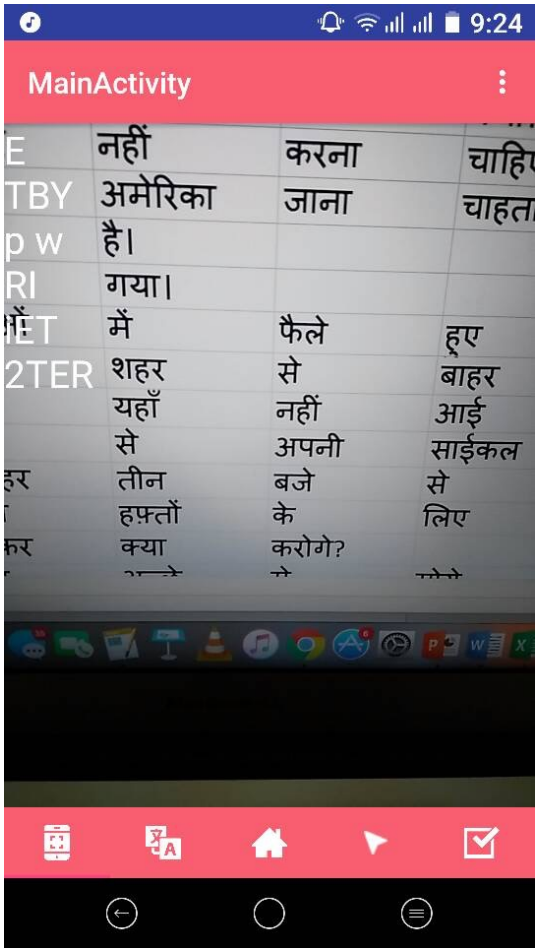




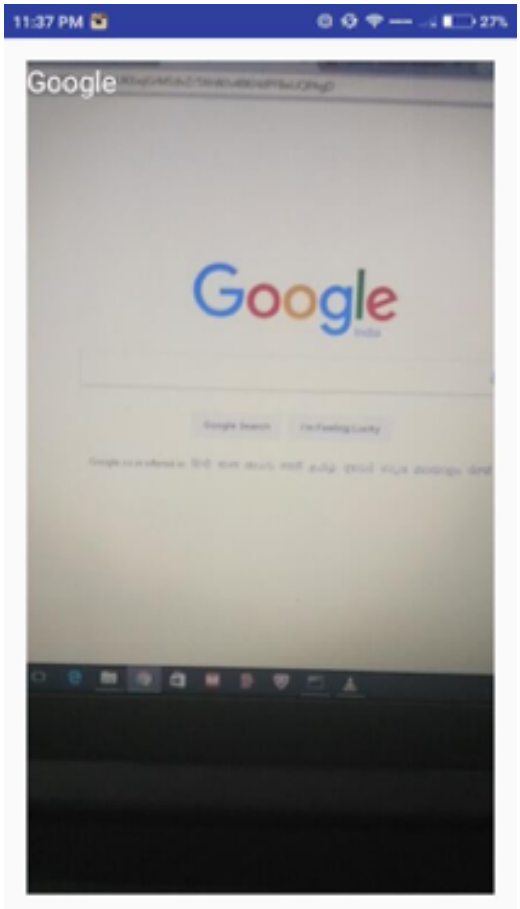
1. Image Translator

First and the main function of the app is the optical translator. It switches the app's view to a camera, and the user is supposed to point the camera to the English text he/she wishes to translate into Hindi.
The English text in the image is converted into text blocks using Optical Character Recognition.
These text blocks are then encoded as vectors and passed into the LSTM which then gives out the corresponding vectors of the translated text in Hindi.





The vectors are then decoded and converted into text blocks which aredisplayed on the screen overlapping the image

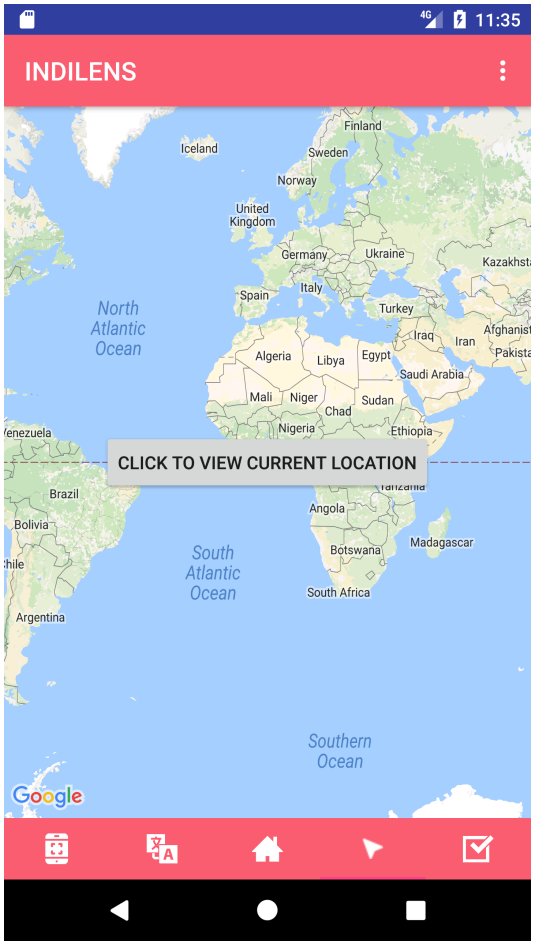


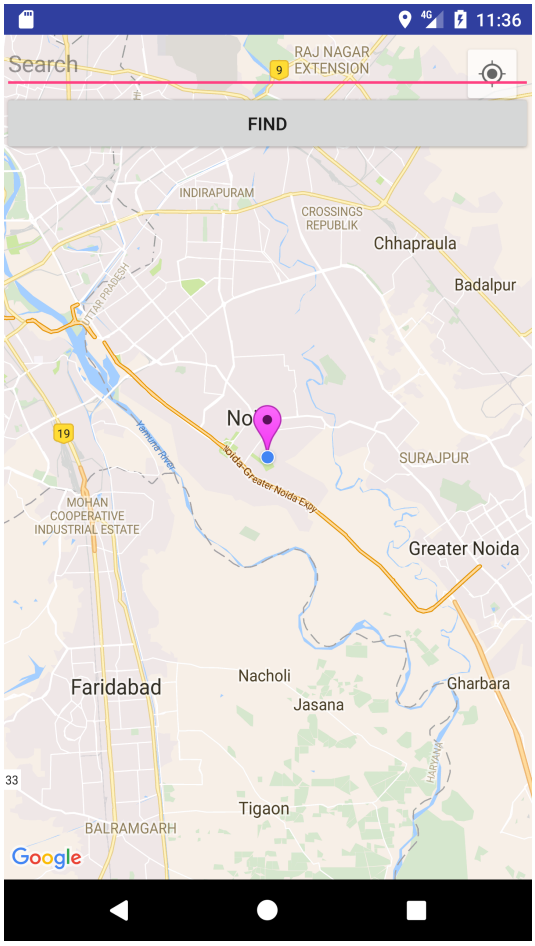
2. Text Translator

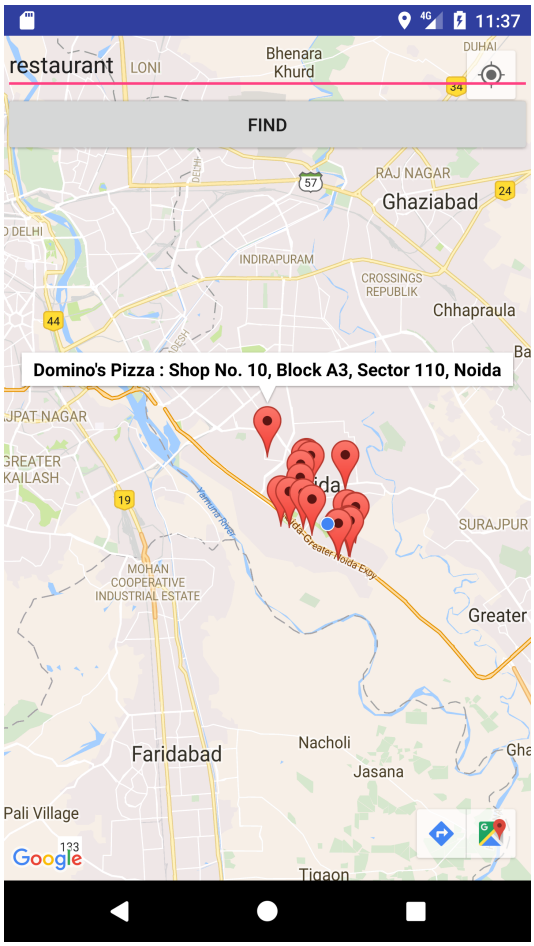
The app also comprises of a text translator from English to Hindi which directly vectorises the text entered by the user, feeds the vector to LSTM which outputs a vector corresponding to the Hindi translation, which is further decoded to display the text to the user.

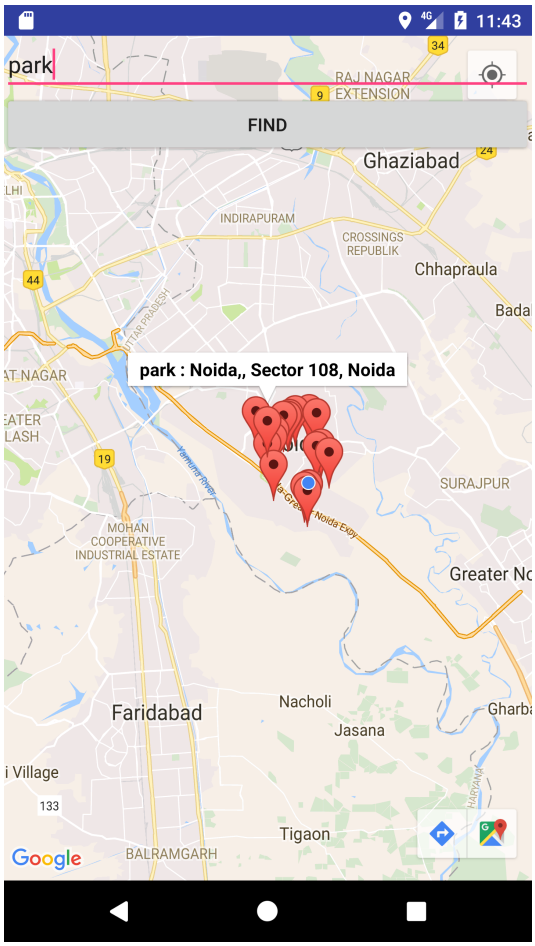
3. Navigator

Other functionalities include integration with the Google Places API which detects the current location of the user and plots it on a map. The user can then search for nearby hotspots like restaurants, atm vestibules, parks, shopping malls etc. by simply typing the ‘place_type’ they wish to search.


11:35
INDILENS
Finland
Iceland
Sweden
Norway
United Kingdom
Germany
Ukraine
Kazakhst
Spain
Italy
Turkey
North Atlantic Ocean
Iraq
Iran
Afghanist
Pakista
Algeria
Libya
Egypt
Saudi Arabia
Mali
Niger
Chad
Sudan
Nigeria
Ethiopia
CLICK TO VIEW CURRENT LOCATION
Brazil
Angola
Bolivia
Botswana
Madagascar
South Atlantic Ocean
South Africa
Argentina
Southern Ocean
Google

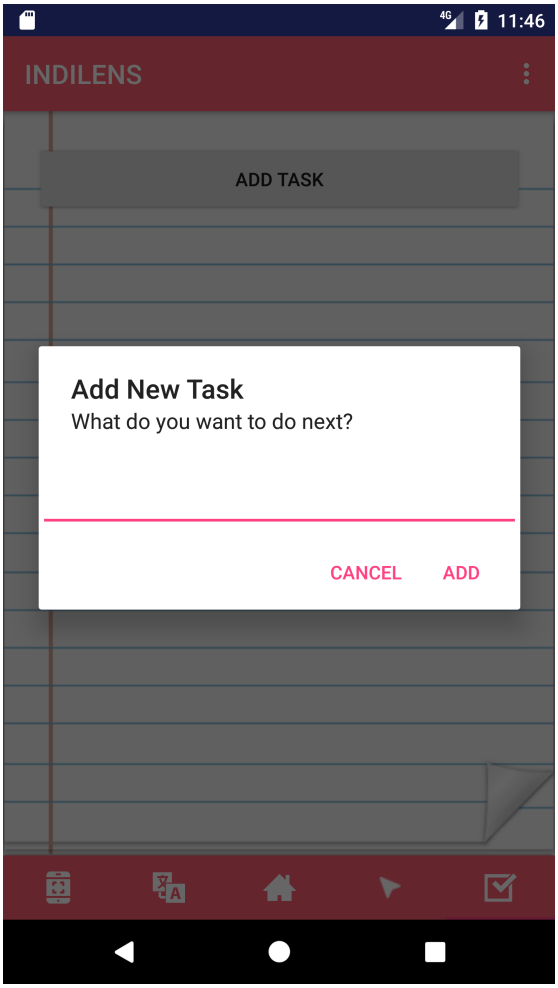

11:36
Search
RAJ NAGAR EXTENSION
FIND
INDIRAPURAM
CROSSINGS REPUBLIK
Chhapraula
Badalpur
SURAJPUR
MOHAN COOPERATIVE INDUSTRIAL ESTATE
Greater Noida
Faridabad
Nacholi
Jasana
Gharbara
Tigaon
BALRAMGARH
Google

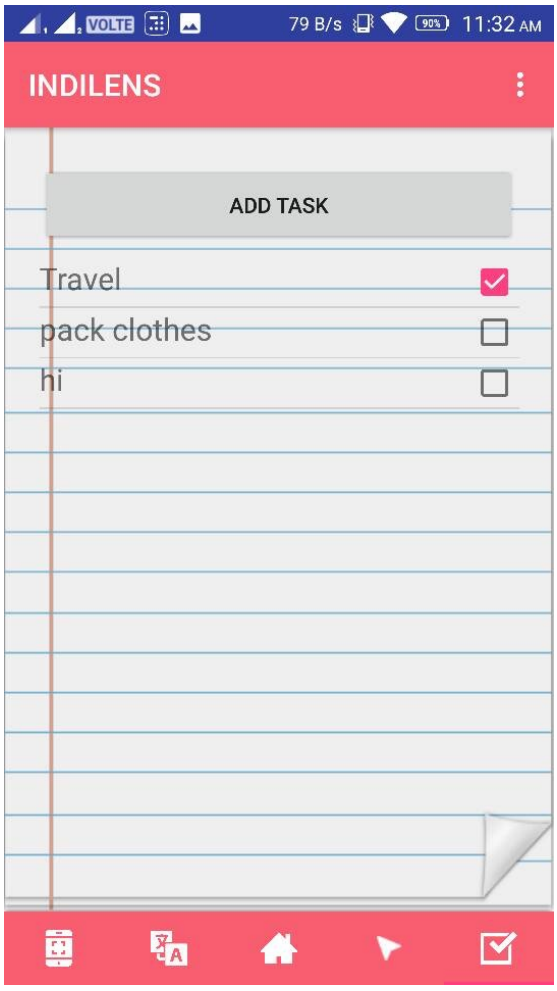

11:37
restaurant
LONI
Bhenara Khurd
FIND
RAJ NAGAR
Ghaziabad
INDIRAPURAM
CROSSINGS REPUBLIK
Chhapraula
Domino's Pizza : Shop No. 10, Block A3, Sector 110, Noida
SURAJPUR
MOHAN COOPERATIVE INDUSTRIAL ESTATE
Greater
Faridabad
Nacholi
Jasana
Pali Village
Tigaon
Google


11:43
park
RAJ NAGAR EXTENSION
FIND
Ghaziabad
INDIRAPURAM
CROSSINGS REPUBLIK
Chhapraula
park : Noida,, Sector 108, Noida
SURAJPUR
MOHAN COOPERATIVE INDUSTRIAL ESTATE
Faridabad
Nacholi
Jasana
Tigaon
BALRAMGARH
Google

4. To-Do List

The last but not the least is the app's in built To-Do List to give the user full ease for his travel experience to make notes, reminders, planners for his trip. To accomplish this SQLite database has been used in Android Studio.





Algorithms being Used

Artificial Intelligence

Artificial Neural Networks are a computational approach which is based on a large collection of neural units loosely modelling the way a biological brain solves problems with large clusters of biological neurons connected by axons. Each neural unit is connected with many others, and links can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together. There may be a threshold function or limiting function on each connection and on the unit itself such that it must surpass it before it can propagate to other neurons. These systems are self-learning and trained rather than explicitly programmed and excel in areas where the solution or feature detection is difficult to express in a traditional computer program.

Recurrent Neural Network (RNN) is the class of artificial neural network where connections between units form a directed cycle. RNNs are very apt for sequence classification problems and the reason they're so good at this is that they're able to retain important data from the previous inputs and use that information to modify the current output.

Our method uses a multi layered Long Short-Term Memory (LSTM) Recurrent Neural Network to map the input sequence to a vector of a fixed dimensionality, and then to decode the target sequence from the vector using KERAS. Our main result is Hindi to English translation.

The idea behind any neural network is that for a provided input the network predicts an output which is then mapped with the expected output. The error is computed for the same and the aim is to minimise it.

The vector conversion is done using Gensim tool (Doc2Vec function) which converts the sentence in an array of numerical values of 1xn dimension where n depends on the input data.

The back conversion will use the reverse algorithm where we convert the vectored output to English. This conversion used the concept of cosine similarity between the output generated and the trained output and then mapping it to the dataset which was used to train the model.

OCR

Optical character recognition is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).

We will be incorporating OCR in our android application using the Mobile Vision Text API.

Text recognition is the process of detecting text in images and video streams and recognizing the text contained therein. Once detected, the recognizer then determines the actual text in each block and segments it into lines and words. The Text API detects text in Latin based languages (French, German, English, etc.), in real-time, on device.

Snapshots

Input sentences vector conversion-

```
LabeledSentence(['मैं', 'मोहन', 'के', 'साथ', 'गेंद', 'खेलने', 'जा', 'रहा', 'हूँ।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [0])
[[-0.00239044 -0.00232981  0.00349048 ...,  0.00088516 -0.0009019
  -0.00362086]
 [ 0.00063952  0.00125202 -0.00281906 ...,  0.00432971  0.00483005
  -0.00133581]
 [-0.0006967   0.00166962  0.00065742 ...,  0.00174471 -0.00387247
  -0.00328605]
 ...,
 [ 0.0014452  -0.00319847 -0.00326479 ..., -0.00155762 -0.0030876
  -0.00119048]
 [ 0.0014452  -0.00319847 -0.00326479 ..., -0.00155762 -0.0030876
  -0.00119048]
 [ 0.0014452  -0.00319847 -0.00326479 ..., -0.00155762 -0.0030876
  -0.00119048]]
LabeledSentence(['तुम', 'अंग्रेज़ी', 'नहीं', 'बोल', 'सकते', 'हो', 'क्या?', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [1])
[[ -2.70895334e-03   2.40082340e-03   1.14890514e-03 ...,  -7.81572191e-04
    4.18101158e-03   1.29756646e-03]
 [  4.94769029e-03  -1.39549689e-03  -3.99999088e-03 ...,  -4.49580140e-03
   -7.13239599e-04  -3.82471271e-03]
 [ -4.40929597e-03   1.16495462e-03   3.75311589e-03 ...,  -9.74765062e-05
    3.41663579e-03   1.79239025e-03]
 ...,
 [  1.45430071e-03  -3.18689435e-03  -3.29005485e-03 ...,  -1.57791434e-03
   -3.11383908e-03  -1.17887405e-03]
 [  1.45430071e-03  -3.18689435e-03  -3.29005485e-03 ...,  -1.57791434e-03
   -3.11383908e-03  -1.17887405e-03]
 [  1.45430071e-03  -3.18689435e-03  -3.29005485e-03 ...,  -1.57791434e-03
   -3.11383908e-03  -1.17887405e-03]]
LabeledSentence(['दुनिया', 'और-भी-और', 'तेज़ी', 'से', 'बदल', 'रही', 'है।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [2])
[[ -3.60308751e-03   4.12198668e-03   1.24334916e-03 ...,   1.70276256e-03
    1.83315855e-03  -1.13255845e-03]
 [  5.92931872e-04   4.29983484e-03  -3.77713190e-03 ...,  -4.03833529e-03
    7.28664978e-04   3.77987861e-03]
 [  1.20922911e-03  -5.88478637e-04   3.49184405e-03 ...,  -4.54349304e-03
    1.44267548e-03  -7.46656515e-05]
 ...,
 [  1.42841728e-03  -3.20954248e-03  -3.28126922e-03 ...,  -1.53850310e-03
   -3.10987118e-03  -1.18077011e-03]
 [  1.42841728e-03  -3.20954248e-03  -3.28126922e-03 ...,  -1.53850310e-03
   -3.10987118e-03  -1.18077011e-03]
 [  1.42841728e-03  -3.20954248e-03  -3.28126922e-03 ...,  -1.53850310e-03
   -3.10987118e-03  -1.18077011e-03]]
LabeledSentence(['आपको', 'अपनी', 'कीमती', 'चीज़ें', 'किसी', 'सुरक्षित', 'जगह', 'पर', 'रखनी', 'चाहिएँ।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [3])
[[ 0.00053141 -0.00493819  0.0042     ...,  0.00162638  0.00255094
  -0.00283239]
```

```
LabeledSentence(['हम', 'उसे', 'नहीं', 'जानते', 'हैं।', '', '', '', '', 'करनी', 'चाहिए।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [27])
[[ -8.29267839e-04  -1.58676121e-03   4.04213276e-03 ...,  -7.74232263e-04
    2.10316805e-03   4.48905909e-03]
 [  1.59238733e-03  -4.30242531e-03   9.32421535e-04 ...,  -1.34993737e-04
   -4.70200699e-04  -1.20108796e-03]
 [ -4.40929597e-03   1.16495462e-03   3.75311589e-03 ...,  -9.74765062e-05
    3.41663579e-03   1.79239025e-03]
 ...,
 [  1.40936940e-03  -3.20817390e-03  -3.26014007e-03 ...,  -1.57415227e-03
   -3.14081158e-03  -1.14917231e-03]
 [  1.40936940e-03  -3.20817390e-03  -3.26014007e-03 ...,  -1.57415227e-03
   -3.14081158e-03  -1.14917231e-03]
 [  1.40936940e-03  -3.20817390e-03  -3.26014007e-03 ...,  -1.57415227e-03
   -3.14081158e-03  -1.14917231e-03]]
LabeledSentence(['उसने', 'हल्के', 'से', 'उसके', 'कंधे', 'पर', 'हाथ', 'रखा।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [28])
[[-0.00074963  0.00362762 -0.00413302 ..., -0.00177626 -0.00418469
  -0.00094236]
 [ 0.00467662  0.00161143 -0.00061055 ..., -0.00093238 -0.00451003
   0.00178185]
 [ 0.00429111 -0.00090573 -0.00446723 ...,  0.00425756  0.00323174
   0.00359521]
 ...,
 [ 0.00144768 -0.00319694 -0.00326765 ..., -0.00156286 -0.0030976
  -0.00116986]
 [ 0.00144768 -0.00319694 -0.00326765 ..., -0.00156286 -0.0030976
  -0.00116986]
 [ 0.00144768 -0.00319694 -0.00326765 ..., -0.00156286 -0.0030976
  -0.00116986]]
LabeledSentence(['मम्मी', 'ने', 'मेरे', 'लिए', 'यह', 'किताब', 'खरीदी।', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''], [29])
[[ 0.00069619 -0.00209677  0.00379405 ..., -0.00459809 -0.00043169
  -0.00244641]
 [-0.00349312  0.00225029  0.00499101 ..., -0.00232523 -0.00323462
   0.00434066]
 [ 0.00412834 -0.00294972  0.00337338 ..., -0.00206602 -0.00150837
  -0.00236527]
 ...,
 [ 0.00145698 -0.00320144 -0.00329227 ..., -0.0015643  -0.00309941
  -0.00115439]
 [ 0.00145698 -0.00320144 -0.00329227 ..., -0.0015643  -0.00309941
  -0.00115439]
 [ 0.00145698 -0.00320144 -0.00329227 ..., -0.0015643  -0.00309941
  -0.00115439]]
model loaded
```

Output sentences vector conversion-

```
LabeledSentence(['I', 'am', 'going', 'to', 'go', 'play', 'ball', 'with', 'Mohan.', '', ''], [0])
[[ -4.25154576e-03  -1.18370482e-03  -1.91180687e-03 ...,  -4.79874806e-03
   -3.77852982e-03  -1.33263544e-04]
 [ -1.38290483e-03  -3.13292793e-03   1.14873191e-03 ...,  -6.91614332e-06
    1.63670571e-03  -4.91250865e-03]
 [  9.25103959e-04   3.12335719e-03  -2.67483317e-03 ...,   3.07788211e-03
   -2.00240221e-03   4.09440091e-03]
 ...,
 [  2.75527663e-03   2.27968831e-05  -1.38464023e-03 ...,   1.66530290e-03
   -7.88812817e-04  -3.33155645e-03]
 [  4.98699583e-03   4.56125243e-03  -4.85171471e-03 ...,   3.75878136e-03
    2.36537494e-03  -4.62688133e-03]
 [  4.98699583e-03   4.56125243e-03  -4.85171471e-03 ...,   3.75878136e-03
    2.36537494e-03  -4.62688133e-03]]
LabeledSentence(['Can', 'you', 'not', 'speak', 'English?', '', '', '', '', '', ''], [1])
[[-0.00095025  0.00137886 -0.00487785 ..., -0.000311   -0.00010533
   0.00214225]
 [-0.00177752  0.00262682  0.00119469 ..., -0.00311262 -0.00395431
  -0.00401035]
 [ 0.00452962 -0.0041567   0.00486481 ...,  0.00406426  0.00107814
  -0.00029195]
 ...,
 [ 0.00498235  0.00456438 -0.00486116 ...,  0.00375102  0.00236925
  -0.00461915]
 [ 0.00498235  0.00456438 -0.00486116 ...,  0.00375102  0.00236925
  -0.00461915]
 [ 0.00498235  0.00456438 -0.00486116 ...,  0.00375102  0.00236925
  -0.00461915]]
LabeledSentence(['The', 'world', 'is', 'changing', 'more', 'and', 'more', 'quickly.', '', '', ''], [2])
[[ 0.0034363  -0.00492897 -0.00241695 ...,  0.00442073 -0.00411332
   0.0028641 ]
 [-0.00122028  0.00207965 -0.00057791 ...,  0.00175763 -0.0018828
  -0.00343554]
 [ 0.00177621 -0.00224423 -0.00044772 ...,  0.00266271  0.0016092
   0.00236121]
 ...,
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]]
LabeledSentence(['You', 'should', 'keep', 'your', 'valuables', 'in', 'a', 'safe', 'place.', '', ''], [3])
[[-0.00228465 -0.00125252  0.00381857 ..., -0.00315736  0.0038983
  -0.00256103]
 [-0.00460596 -0.00078259 -0.00063194 ..., -0.00332759  0.00191275
  -0.00474397]
 [ 0.00343095 -0.00160069 -0.00411569 ...,  0.00155415 -0.00334079
  -0.00103363]
```

```
LabeledSentence(['We', "don't", 'know', 'him', '', '', '', '', '', '', ''], [27])
[[ 0.00416599  0.00379023  0.00366376 ..., -0.00151419 -0.001347
  -0.00469248]
 [ 0.00333892  0.00369347  0.00144487 ..., -0.00093775  0.00266404
  -0.00364813]
 [ 0.00469931  0.00394975 -0.0026822  ...,  0.00479952  0.00300536
   0.00217097]
 ...,
 [ 0.00499013  0.00456683 -0.00484681 ...,  0.00375482  0.00236621
  -0.00462762]
 [ 0.00499013  0.00456683 -0.00484681 ...,  0.00375482  0.00236621
  -0.00462762]
 [ 0.00499013  0.00456683 -0.00484681 ...,  0.00375482  0.00236621
  -0.00462762]]
LabeledSentence(['She', 'put', 'her', 'hands', 'gently', 'on', 'his', 'shoulders.', '', '', ''], [28])
[[ 0.00198927  0.00251865  0.00490687 ..., -0.00130747  0.00041493
  -0.00356128]
 [-0.00141806  0.00335144 -0.00414176 ...,  0.00214219 -0.00229752
   0.0045927 ]
 [-0.00478994 -0.0033231   0.00096165 ..., -0.00083285 -0.00272965
  -0.00408822]
 ...,
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]
 [ 0.004987    0.00456125 -0.00485171 ...,  0.00375878  0.00236537
  -0.00462688]]
LabeledSentence(['Mother', 'bought', 'me', 'the', 'book.', '', '', '', '', '', ''], [29])
[[-0.00414622 -0.00495316  0.00254239 ...,  0.00492398 -0.00072593
  -0.00373455]
 [ 0.00458816  0.00416252  0.00031499 ...,  0.00358211  0.00366978
   0.00352368]
 [-0.00209059  0.00049914 -0.00168705 ..., -0.00188199  0.00201285
  -0.00093509]
 ...,
 [ 0.00498581  0.00456544 -0.00486093 ...,  0.00376712  0.00236749
  -0.0046281 ]
 [ 0.00498581  0.00456544 -0.00486093 ...,  0.00376712  0.00236749
  -0.0046281 ]
 [ 0.00498581  0.00456544 -0.00486093 ...,  0.00376712  0.00236749
  -0.0046281 ]]
```

Training LSTM for 500 sentences (input: english, output: hindi)

```
Using TensorFlow backend.
model loaded
12
100
(None, 12, 100)
(None, 12, 100)
Epoch 1/150

499/499 [==============================] - 1s - loss: -3.0458e-04 - acc: 1.6700e-04
Epoch 2/150

499/499 [==============================] - 0s - loss: -6.2589e-04 - acc: 5.0100e-04
Epoch 3/150

499/499 [==============================] - 0s - loss: -9.4356e-04 - acc: 3.3400e-04
Epoch 4/150

499/499 [==============================] - 0s - loss: -0.0013 - acc: 0.0000e+00
Epoch 5/150

499/499 [==============================] - 0s - loss: -0.0016 - acc: 1.6700e-04
Epoch 6/150

499/499 [==============================] - 0s - loss: -0.0019 - acc: 8.3500e-04
Epoch 7/150

499/499 [==============================] - 0s - loss: -0.0022 - acc: 0.0010
Epoch 8/150

499/499 [==============================] - 0s - loss: -0.0025 - acc: 0.0013
Epoch 9/150

499/499 [==============================] - 0s - loss: -0.0028 - acc: 0.0013
Epoch 10/150

499/499 [==============================] - 0s - loss: -0.0031 - acc: 0.0018
Epoch 11/150

499/499 [==============================] - 0s - loss: -0.0033 - acc: 0.0012
Epoch 12/150

499/499 [==============================] - 0s - loss: -0.0036 - acc: 0.0023
Epoch 13/150

499/499 [==============================] - 0s - loss: -0.0039 - acc: 0.0012
Epoch 14/150
499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4095
Epoch 128/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4060
Epoch 129/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4140
Epoch 130/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4026
Epoch 131/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4093
Epoch 132/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4167
Epoch 133/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.3991
Epoch 134/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4185
Epoch 135/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4188
Epoch 136/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4207
Epoch 137/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4232
Epoch 138/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4369
Epoch 139/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4275
Epoch 140/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4332
Epoch 141/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4389
Epoch 142/150

499/499 [==============================] - 0s - loss: -0.0070 - acc: 0.4329
Epoch 143/150
```

```
 1/499 [..............................] - ETA: 39s
 2/499 [..............................] - ETA: 54s
 3/499 [..............................] - ETA: 58s
 4/499 [..............................] - ETA: 64s
 5/499 [..............................] - ETA: 68s
 6/499 [..............................] - ETA: 73s
 7/499 [..............................] - ETA: 74s
 8/499 [..............................] - ETA: 77s
 9/499 [..............................] - ETA: 79s
10/499 [..............................] - ETA: 82s
11/499 [..............................] - ETA: 84s
12/499 [..............................] - ETA: 88s
13/499 [..............................] - ETA: 90s
14/499 [..............................] - ETA: 93s
15/499 [..............................] - ETA: 97s
16/499 [..............................] - ETA: 100s
17/499 [>.............................] - ETA: 103s
18/499 [>.............................] - ETA: 106s
19/499 [>.............................] - ETA: 110s
20/499 [>.............................] - ETA: 114s
21/499 [>.............................] - ETA: 119s
22/499 [>.............................] - ETA: 124s
23/499 [>.............................] - ETA: 130s
24/499 [>.............................] - ETA: 136s
```

```
319/499 [==================>...........] - ETA: 3111s
320/499 [==================>...........] - ETA: 3120s
321/499 [==================>...........] - ETA: 3131s
322/499 [==================>...........] - ETA: 3135s
323/499 [==================>...........] - ETA: 3141s
324/499 [==================>...........] - ETA: 3150s
325/499 [==================>...........] - ETA: 3151s
326/499 [==================>...........] - ETA: 3158s
327/499 [==================>...........] - ETA: 3166s
328/499 [==================>...........] - ETA: 3177s
329/499 [==================>...........] - ETA: 3186s
330/499 [==================>...........] - ETA: 3194s
331/499 [==================>...........] - ETA: 3194s
332/499 [==================>...........] - ETA: 3201s
333/499 [===================>..........] - ETA: 3205s
334/499 [===================>..........] - ETA: 3208s
335/499 [===================>..........] - ETA: 3208s
336/499 [===================>..........] - ETA: 3214s
337/499 [===================>..........] - ETA: 3213s
338/499 [===================>..........] - ETA: 3220s
339/499 [===================>..........] - ETA: 3224s
340/499 [===================>..........] - ETA: 3224s
341/499 [===================>..........] - ETA: 3226s
342/499 [===================>..........] - ETA: 3228s
```

Output of the LSTM for given input

```
[array([ 0.06716418], dtype=float32), array([ 0.06451652], dtype=float32), array([ 0.06741673], dtype=float32), array([ 0.06720158], dtype=float32), array([ 0.0665436], dtype=float32), array([ 0.0652399
], dtype=float32), array([ 0.06633027], dtype=float32), array([ 0.06709965], dtype=float32), array([ 0.06727916], dtype=float32), array([ 0.06728371], dtype=float32)]
['The','heavy','rain','brought','the','flood','causing','a' ,'lot','of','damage','around.']
['दुनिया', 'और-भी-और', 'तेज़ी', 'से', 'बदल', 'रही', 'है।', '', '', '', '', '']
>>>
```

References

- https://keras.io/
- https://radimrehurek.com/gensim/models/doc2vec.html
- https://www.tensorflow.org/get_started/get_started
- https://www.youtube.com/watch?v=QAbQgLGKd3Y&list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl
- https://www.youtube.com/watch?v=VINCQghQRuM
- https://developers.google.com/vision/text-overview
- https://codelabs.developers.google.com/codelabs/mobile-vision-ocr/#0
- https://thenewboston.com/videos.php?cat=278
- https://developers.google.com/places/android-api/start
- www.stackoverflow.com