

LANGUAGE IDENTIFICATION AND TRANSLATION USING NEURAL NETWORKS

**A Project Report submitted in partial fulfilment of the requirements for the award of the
degree of**

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

| | |
|-------------------------|---------------------|
| D.V. ANAND SAI | 121710303011 |
| K. ROHITH | 121710303025 |
| P. CHARITH REDDY | 121710303042 |

Under the esteemed guidance of

Dr. JHANSI YELLAPU

Assistant Professor, CSE Department, GIT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

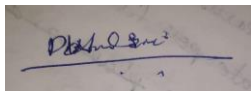
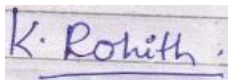
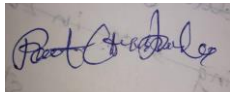
(Deemed to be University)



DECLARATION

We, hereby declare that the Project review entitled “**LANGUAGE IDENTIFICATION AND TRANSLATION USING NEURAL NETWORKS**” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

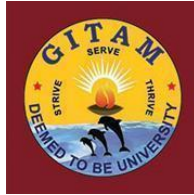
Date: 11th May, 2021.

| Registration No: | Name: | Signature: |
|------------------|------------------|---|
| 121710303011 | D.V. ANAND SAI |  |
| 121710303025 | K. ROHITH |  |
| 121710303042 | P. CHARITH REDDY |  |

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM (Deemed to be University)



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “**LANGUAGE IDENTIFICATION AND TRANSLATION USING NEURAL NETWORKS**” is a bonafide record of work carried out by **D.V. ANAND SAI(121710303011),K.ROHITH(121710303025),P. CHARITH REDDY (121710303042)** submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

PROJECT GUIDE

Dr. JHANSI YELLAPU
Assistant Professor
CSE Department, GIT
GITAM

HEAD OF THE DEPARTMENT

Dr.R. SIREESHA
Professor
CSE, GIT
GITAM

ACKNOWLEDGEMENT

We would like to thank our project guide **Dr. Jhansi Yellapu**, Assistant Professor, Department of CSE for her stimulating guidance and profuse assistance. We shall always cherish our association with her for her guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under her guidance and constant support.

We also express our thanks to the project's reviewers **Dr. Srinivas G**, Associate Professor, **Bhargav K**, Assistant Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance for doing our project.

We consider it is a privilege to express our deepest gratitude to **Dr. R. Sireesha**, Head of the Department, Computer Science Engineering for her valuable suggestions and constant motivation that greatly helped us to successfully complete this project.

Our sincere thanks to **Dr. C. Dharma Raj**, Principal, GITAM Institute of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

D.V. ANAND SAI **121710303011**

K. ROHITH **121710303025**

P. CHARITH REDDY **121710303042**

ABSTRACT

With the coming of Deep learning numerous new Language Translation Methods (LTM) have been conceivable with regards to Natural Language Processing (NLP). One such method is the Language translation utilizing neural networks in Deep Learning (DL). To this method to be carried out the language of the content or sound must be recognized which is done manually. We might want to propose a system where we merge the Language Identification System (LID) with the translation framework in order to recognize the language and input the equivalent to the translation framework. To carry out this LID, we utilize the methodology of Long Short-Term Memory (LSTM) , a sort of Recurrent Neural Network. The input will be given as text and will be distinguished which language it is and afterward be translated into the language needed. These LSTM within the RNN framework have shown promising results in many of the researches done by other Research Groups when compared to that of the Machine Learning methodologies like Naive Bayes, Support Vector Machines and so on. To build these Deep Learning frameworks we will utilize built-in frameworks like numpy, pandas, matplotlib, NLTK for Natural Language Processing, TensorFlow and Keras for carrying out DL methodologies in the Python programming language. The Integrated Development Environment (IDE) which we use will be Jupyter notebook and Spyder within the Anaconda environment for Version control. At last we will create a front end where the end client could give the input for the purpose of translation.

TABLE OF CONTENTS

| CONTENTS | PAGE.NO |
|--|----------------|
| Declaration | i |
| Certificate | ii |
| Acknowledgement | iii |
| Abstract | iv |
| 1. Introduction | 1 |
| 1.1 - Deep Learning | 1 |
| 1.2 - Natural Language Processing | 2 |
| 2. Literature Survey | 5 |
| 3. Problem Identification & Objectives | 6 |
| 3.1 - Language Identification and Translation Problem | 6 |
| 3.2 - Objectives and Proposed Solution | 6 |
| 4. System Design | 10 |
| 4.1 - User Perspective | 10 |
| 4.2 - System Working from Developer's perspective | 11 |
| 4.3 - System Implementation from Developer's perspective | 12 |
| 5. Methodology | 14 |
| 5.1 -Platforms/IDE's | 14 |
| 5.2 -Libraries | 14 |
| 5.3 -NLP Methods | 15 |
| 6. Implementation | 16 |
| 6.1 -Datasets | 16 |
| 6.2 -Language Translation | 17 |
| 6.3 -Language Identification | 20 |
| 6.4 -Integration of Both | 23 |
| 6.5 -Front End | 24 |
| 7. System Testing | 27 |
| 7.1 - Language Translation Testing | 27 |
| 7.2 - Language Identification Testing | 28 |
| 8. Results | 29 |
| 9. Conclusion | 31 |

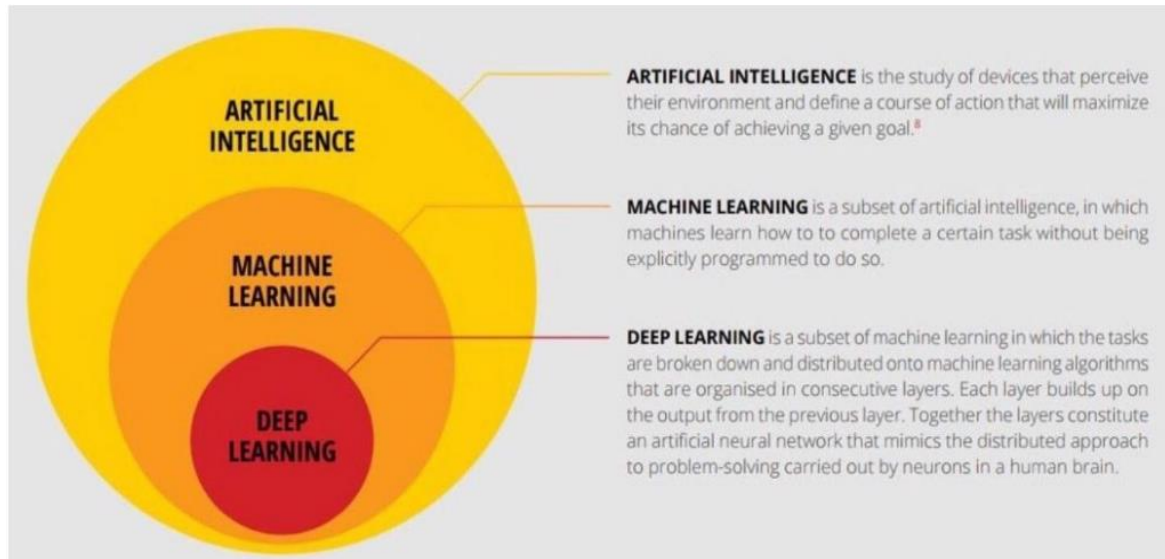
LIST OF FIGURES

| Figure No. | Description | Page No. |
|-------------------|--|-----------------|
| 4.1 | User Perspective . | 10 |
| 4.2 | System Working from Developer's perspective | 11 |
| 4.3 | System Implementation from Developer's perspective | 12 |

1.Introduction

1.1 Deep Learning:

Deep Learning like Machine Learning (ML) is a kind of Artificial Intelligence (AI). AI is a general term that suggests philosophies that engage PCs to perform human-like activities. Computer based intelligence tends to solve a lot of issues with the assistance of techniques like Artificial Neural Networks.



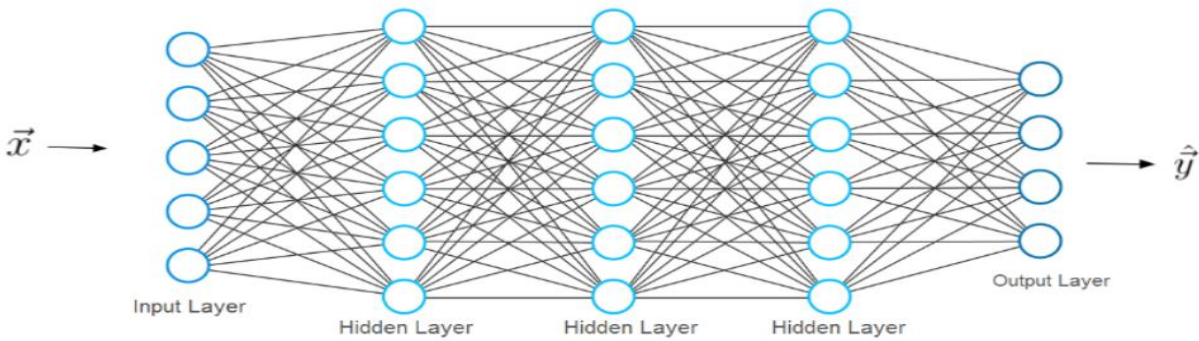
The concept of Deep Learning

Deep Learning is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analysing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.

The plan of the neural networks depends on the design of the human brain. Similarly as we think carefully to recognize designs and characterize various sorts of data, neural networks also play similar roles on information. Whenever we get new data, the brain attempts to distinguish it with the known items. A similar idea is likewise utilized by Deep Neural Networks.

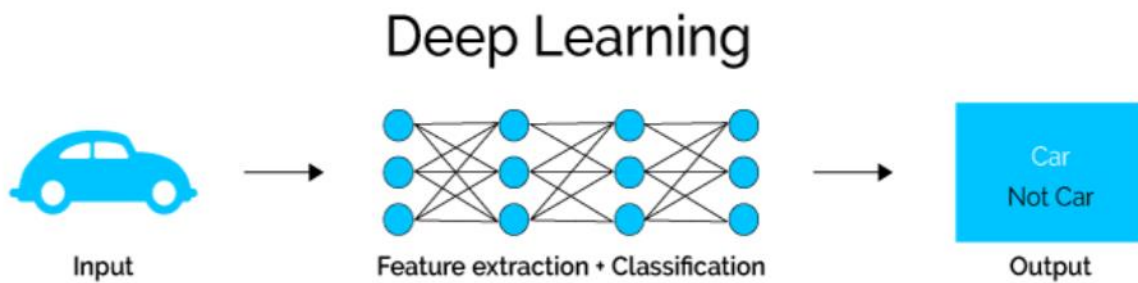
Deep learning utilizes Deep intuitive ways to deal with obvious results dependent upon input information. By and large, Deep learning is unaided. Deep learning depends upon depiction. Possibly than utilizing techniques that are task based. The Deep Learning models mostly correspond to the usage of neural networks.

Neural networks are a bunch of methodologies, inspired from the human brain, that are intended to perceive designs. They work on input information through an intuitive understanding of the given data. The input information is converted to a set of mathematical designs, contained in vectors, into which real world information, be it pictures, sound, text has to be inferred to get intuitive results.



Structure of a Neural Network

To utilize an AI model to decide whether a specific picture is showing a vehicle or not, we people first need to distinguish the properties or attributes of a vehicle like shape, size, windows, wheels, and so on to separate the attributes individually and offer them to the model as information. The model would perceive these attributes of a vehicle and give the right output.



1.2 Natural Language Processing:

Natural Language Processing is the field of AI that is based on making Computers to gain an intuition of the normal languages like Telugu, English, etc. With the assistance of NLP, one could develop the methodologies to make Computers understand normal language. It has a lot of utilities these days.

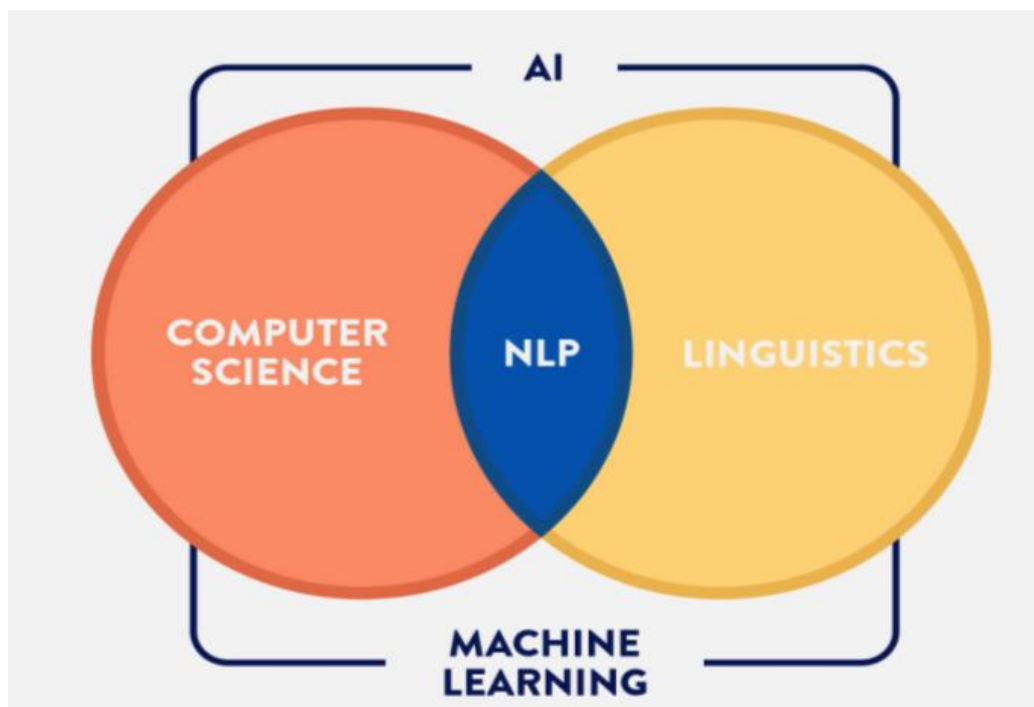
It is one of the sub-topics that centers around the communication between data science and natural language, and is already utilized by innumerable businesses. Today, NLP is booming as a result of enormous upgrades in computational force alongside the availability of huge amounts of information.

NLP has two categories:

1. Natural Language Understanding (NLU).
2. Natural Language Generation (NLG).

NLP is the overlapping of the following two different domains:

- a) The Linguistics: It is concerned with language, it's structure, grammar, meaning, different kinds of phrases and so on.
- b) The Computer Science: It is concerned with applying linguistics, by converting it into mathematical designs that are understandable to the AI models so as to work upon and produce intuitive results.



Essence of Natural Language Processing

NLP involves the research to bring PCs closer to a human-level perception of regular language. Computers don't yet have the exceptionally amazing perception of regular language that people have. A lot of research is underway in this field as of right now.

The main theme of Natural Language Processing lies in causing PCs to comprehend the characteristic language. That is not a simple task however. PCs can comprehend the organized type of information like bookkeeping pages and the tables in the data set, however natural languages, messages, and voices structure make an unstructured classification of information, and it gets hard for the PC to get it, and there emerges the requirement for Natural Language Processing.

Uses of NLP on a daily basis:

- NLP empowers the acknowledgment and forecast of infections dependent on electronic health records and the patient's own experience. Amazon Comprehend Medical is a help that utilizes NLP to gain insight into the sickness conditions, medical and therapy results from patient notes, clinical reports and other electronic health records.
- Companies can figure out what the clients are insisting about a product or an item by distinguishing and extracting information from sources like social media. This is called a Sentimental Analysis.
- An employee at IBM built up an intellectual aide that works like a customized internet searcher by learning about you and afterwards reminding you anything you can't recollect the second you need it to. This is called a Cognitive Assistant.
- Organizations like Yahoo and Google channel and arrange your messages with NLP by breaking down text in messages that move through their servers and halting spam before they even enter your inbox.
- To assist with distinguishing fake news, the NLP Research Group at MIT built up a framework to decide whether a source is a good one or politically one-sided, identifying if a news source can be trusted or not.
- Amazon's Alexa and Apple's Siri are instances of smart voice interfaces that utilize NLP to react to vocal prompts and do all that like tracking down a specific shop, giving weather information, recommending the best course to the workplace, or turning on the lights at home.
- Controlled by IBM Watson NLP Research Group, LegalMation built up a stage to robotize routine prosecutions and help law groups save time and drive down expenses.
- NLP can be utilized to classify text, generate records, allocate ordering and looking for plagiarised content in tremendous volumes of text or oral information otherwise called Categorization.
- Content Identification in NLP can possibly distinguish the significance and bigger subjects inside huge volumes of text information.

2. Literature Survey:

We referred to the research paper on **“Language Identification Using Deep Convolutional Recurrent Neural Networks”** which is published by Christian Bartz, Tom Herold, Haojin Yang and Christoph Meinel from the Hasso Plattner Institute, University of Potsdam. In their research they have used a hybrid Convolutional Recurrent Neural Network (CRNN) to identify the language from the inputs given in image format. In their research they have used the Convolutional Neural Networks (CNN's) to analyse the spatial information and RNN's to identify the language by capturing the information extracted by the CNN's. We have implemented the RNN model in our project using the concept of identity vectors or the i - vectors concept taken from **“Automatic Language Identification Using Deep Neural Networks”** an IEEE research paper.

We have considered the usage of RNN because RNN model in LSTM approach has given very accurate results when compared to that of the other approaches such as the Multinomial Naive Bayes and Logistic Regression as stated by Priyank Mathur, Arkajyothi Misra and Emrah Budur in their paper titled **“Language Identification from Text Documents”**.

Even the problem of vanishing and exploding gradients posed by the RNN could be solved using the Long Short Term Memory (LSTM). Thus, with LSTM's help we would be able to develop a pretty good Language Identification system and the same could be used for the translation purposes.

We have considered the RNN because we wanted to develop a translation system where the input language could be identified by the system beforehand and could be translated. As well as the input which will be given would be in the format of the text.

3. Problem identification and objectives:

3.1 language Identification and Translation Problem:

The language identification (LID) is a technique of determining the given language input in various formats like text, speech, video and so on. The problem corresponds to the automatic language detection during translation of the languages where the user has to specify the language in which the input will be given. Instead of that we propose a combination of this language identification and translation models where the given input would be first monitored for the identification of the language and then it would be translated. At first we would give the input in the form of a text.

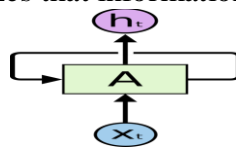
There are several existing API (Application Programming Interfaces) in Python that are the interfaces of the Google translator such as the Goslate, Google Translate and so on. We have chosen Goslate and compare the results between our model and those of the Goslate. The Goslate module could be installed from PyPI using the pip or just could be downloaded as goslate.py file and used directly.

The language translation model is also built on a similar notion. The language translation would be done once the language itself has been identified. Thus at first the Identification model is run and then the Translation model is executed. But in implementation the translation model has to be built first so as to understand the dynamics of passing the parameters to the same from the Identification model.

3.2 Objectives and Proposed Solution:

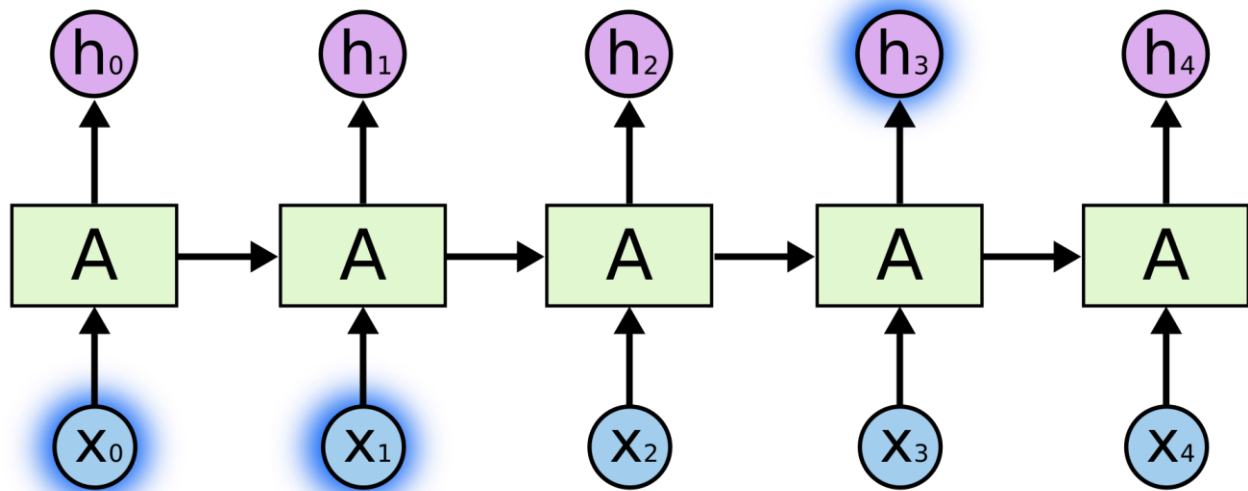
The Recurrent Neural Networks (RNN's) are a part of the Artificial Neural Networks (ANN's) and the Deep Recurrent Neural Networks (DNN's) are specialised forms of RNN's. They come under the category of Feedforward Neural Networks and exhibit the property of temporal dynamic behaviour by counting in the previous nodes output as the input to the current node. There are kinds of RNN's like the Long Short-Term Memory (LSTM), Bi - Gated Recurrent Units (GRU's) and so on. The LSTM has a widespread usage when it comes to Natural Language Processing (NLP) techniques such as translation of the languages, language identification, multilingual language processing.

Recurrent Neural Network recollects the past and it's choices are impacted by knowledge it has gained from the past. Basic feed forward networks recall things as well, however they recollect things they got to know during training. For instance, a picture classifier realizes what a '1' looks like during training and afterward utilizes that information to group things underway.



Structure of Neuron in an RNN

RNN's take at least one input vector and produce at least one output vectors while considering the past neuron's output vectors as the contributions to the current one alongside the loads.



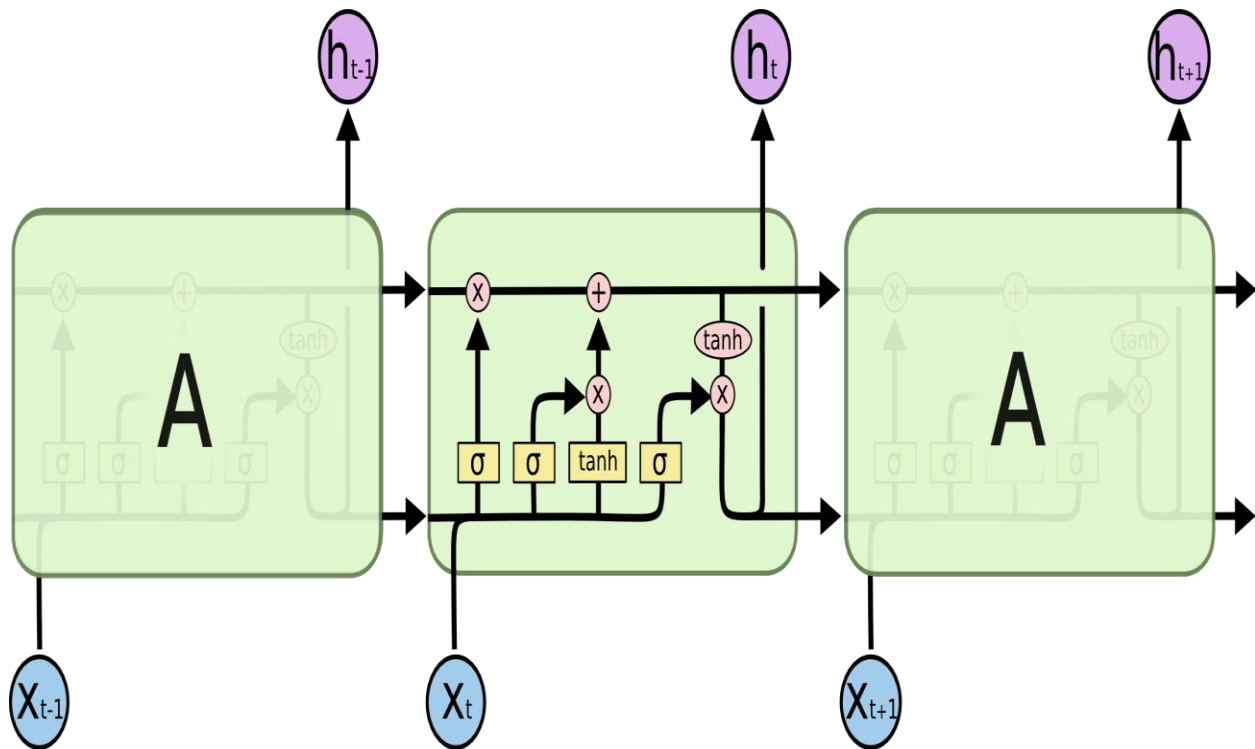
Structure of an RNN

Consider attempting to foresee the final say regarding the content "I experienced childhood in France... I speak fluent French." Recent data proposes that the following word is presumably the name of a language, however on the off chance that we need to limit which language, we need the setting of France, from further back. It's completely feasible for the gap between the important data and where it is expected to turn out to be enormous. This is known as a long term dependency issue.

All Recurrent Neural Networks have the type of a chain of recurring modules of neural networks. In standard RNNs, this recurring module will have a basic design, for example, a solitary tanh layer.

LSTM networks are an augmentation of Recurrent Neural Networks (RNNs) mainly acquainted with handling circumstances where RNNs are not feasible. Discussing RNN, it is an organization that deals with the current input information by looking over the past output information and putting it away in its memory for a brief timeframe. Out of its different applications, the most famous ones are in the fields of speech processing, non-Markovian control and music composition. The disadvantages of RNNs are negligence to store data for a more extended timeframe. Now and again, a reference to certain data put away for a significant time frame is needed to foresee the current output. Yet, RNNs are totally unequipped for dealing with such "long term problems". Second, there could be no better authority over what part of the sentence or input should be conveyed and what part has to be left.

LSTMs intended to stay away from the long term dependency issue. Recalling data for extensive stretches of time is basically their default property.



Structure of LSTM

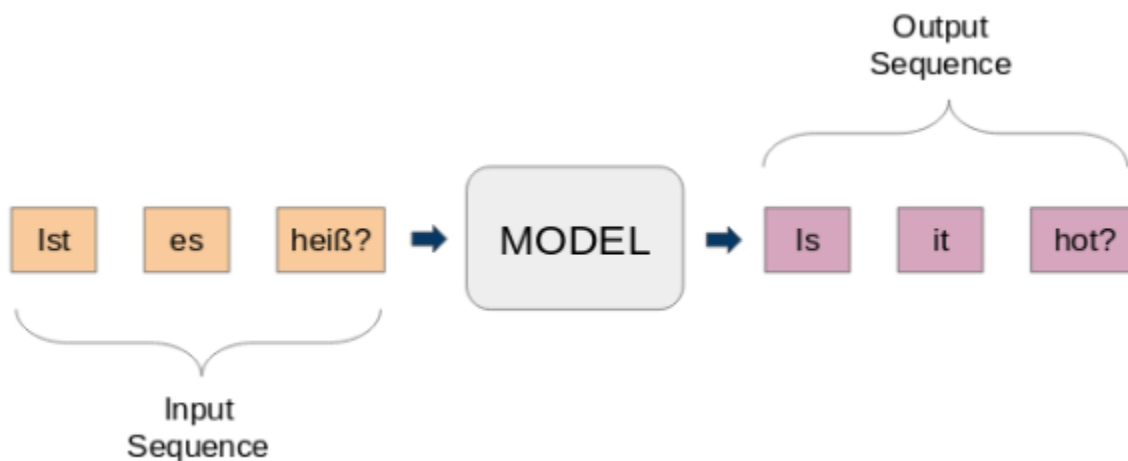
Joining the RNN with LSTM yielded more noteworthy outcomes while distinguishing languages from the text inputs. The central idea of LSTM is the cell state, and it's different gates. The cell state goes about as a vehicle that moves relative data right down the network chain. You can consider it the "memory" of the network. The cell state, in general, can yield important data all throughout the running of the model. So even data from the past steps can move toward later time steps, lessening the impacts of short term memory. As the cell state goes on, data gets added or eliminated to the cell state by means of gates. The gates are diverse neural networks that choose which data is permitted on the cell state. The gates can realize what data is good to keep or forget during training.

LSTMs have an advantage over ordinary feed-forward neural networks and RNN from various perspectives. This is a direct result of their property of specifically recalling information over long time frames.

The essential contrast between the structures of RNNs and LSTMs is that the secret layer of LSTM is a gated unit or gated cell. It comprises four layers that collaborate with each other in order to create the output of that cell alongside the cell state. Not at all like RNNs which have the solitary single neural network layer of tanh, LSTMs contain three sigmoid gates and one tanh layer. Gates have been combined all together to limit the data that is gone through the cell. They figure out what part of the data will be required by the following cell and what part is to be discarded. The output is normally in the scope of 0-1 where '0' signifies 'reject all' and '1' signifies 'accept all'.

Sequence-to-Sequence models are utilized over a large number of NLP methodologies, like text summarizing, speech recognition and so on. Our point is to get an intuitive understanding of given sentences starting with one language then onto the next.

Here the input and output sequences correspond to the sentences that are input by the user and the resultant output respectively.



Sequence to Sequence model encoding in RNN

A normal seq2seq model has 2 significant segments. They are:

- a) Encoder pursues the information pattern and sums up the data in something many refer to as the interior state vectors or setting vector (if there should be an occurrence of LSTM these are known as the context vectors). We dispose of the outputs of the encoder and just protect the inside states. This context vector expects to categorize the data for all information components to help the decoder make exact forecasts.
- b) The decoder is an LSTM whose underlying states are set to the last conditions of the Encoder LSTM, for example the context vector of the encoder's last cell is contribution to the main cell of the decoder network. Utilizing these underlying states, the decoder begins producing the output succession, and these outputs are likewise memorized over for future outputs.

Both the decoder and encoder networks are importantly two varying Recurrent Neural Network (RNN) models agglomerated to form one big network. At times they can be two LSTMs or any others like GRUs (Gated Recurrent Units) etc.

4. Flowchart:

4.1 User Perspective:

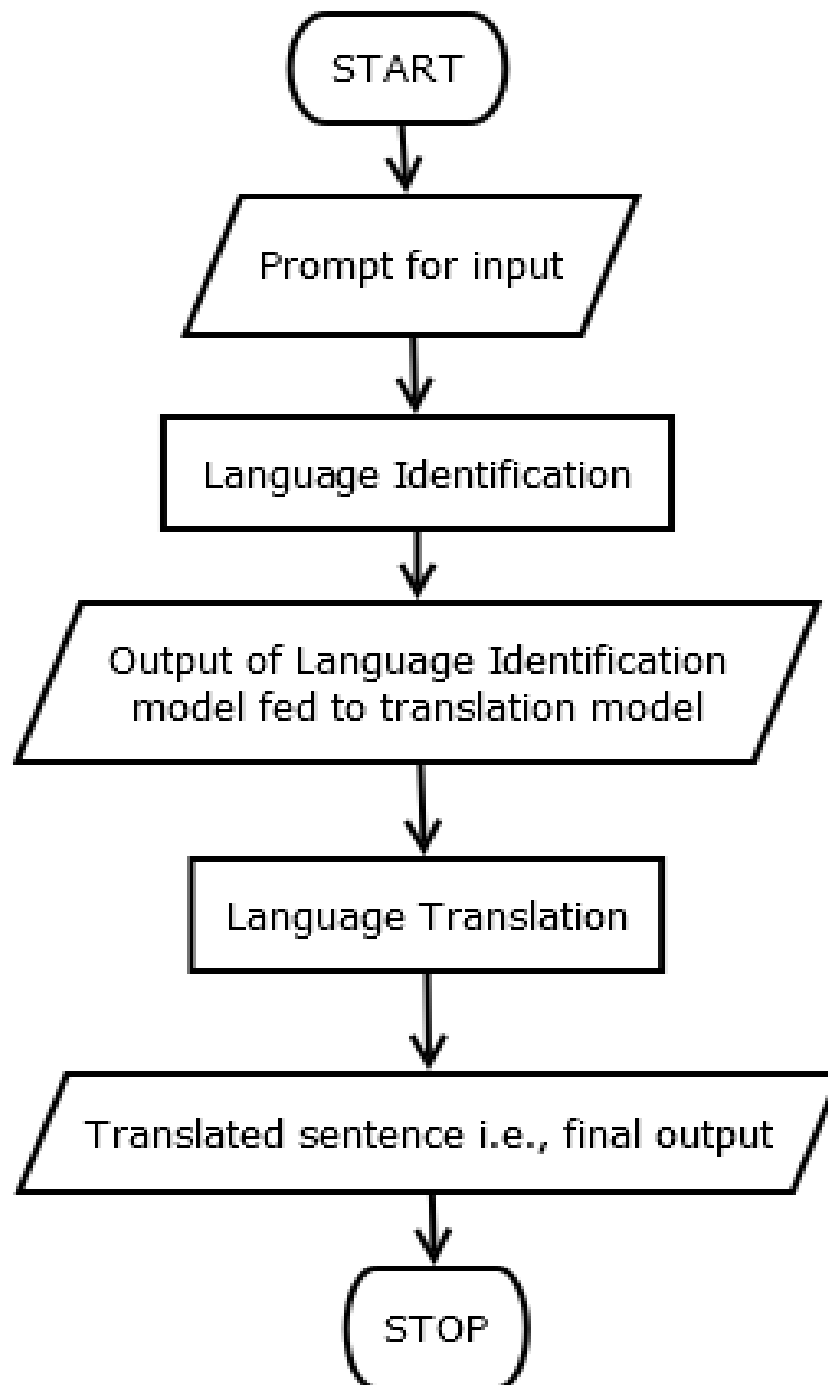


Diagram-1

User Perspective

4.2 System Working from developer's perspective:

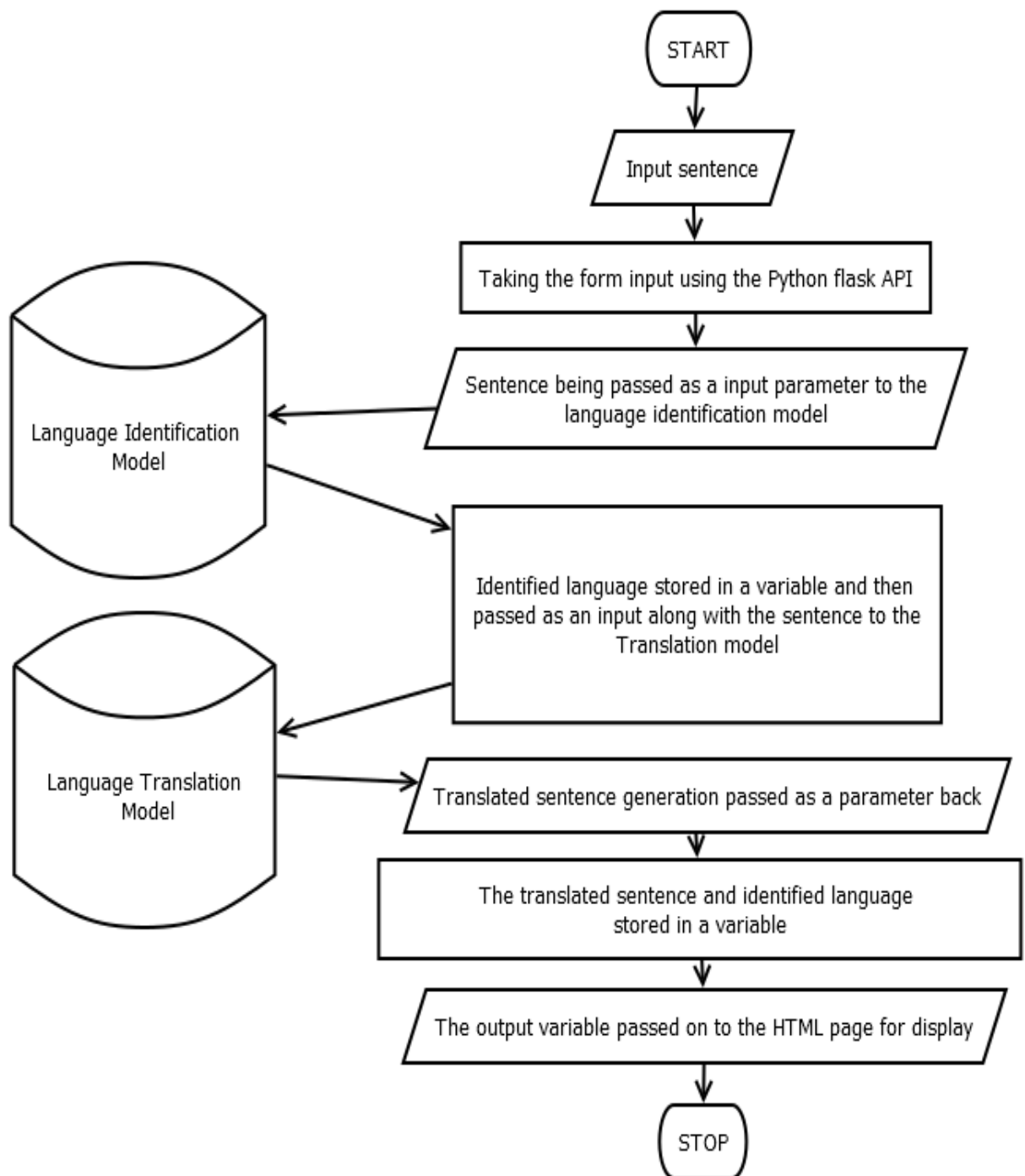


Diagram-2

System Working from developer's perspective

4.3 System Implementation from Developer's perspective:

Language ID is the initial phase in any content evaluation or natural language processing pipeline. On the off chance that the language of a record is misidentified, all ensuing language models will deliver off base outcomes. Blunders at this phase can bring about invalid outcomes, as when an English language analyzer is applied to German content. It is imperative to recognize the language of each sentence created from input.

Machine Translation is the interpretation of text by a PC with no human contribution. Neural networks have taken a jump forward to machine interpretation. This task includes the structure of a non-contiguous neural network that accompanies from the start to finish of the interpretation pipeline. The finished pipeline would acknowledge German language interactive text input. The yield would be produced in the English language.

The i-vectors concept has been used for the purpose of translation model to translate the input. The i-vectors are nothing but the features extracted by the previous RNN model. These will be in the format of single column and multiple row vectors that would contain the numerals corresponding to certain features of the given input.

The quantity of epochs to be run is just controlled by taking a glance at the validation loss and the training loss. In principle the training loss should be diminished such that it ought to become '0' while the validation loss at a point would be at its absolute bottom and in this manner takes a vertical bend that is it would build to an ever increasing extent.

In this manner there would be a specific point at where the previously mentioned uniqueness starts. That is where one requires to quit running the quantity of epochs else it would influence a huge blunder in the model by driving it to overfitting.

Hypothetically, the validation loss would get 100%. In this manner with the training loss being '0%' and the validation loss being 100% the model is said to enter a phase called overfitting where the model could precisely anticipate a 100% on the training data and would foresee predictions over testing data.

The model would be supposed to be underfit if the number of epochs ran would be lesser than the unique point. Subsequently when a model has the quantity of epochs approached the disparity, the model would be a good one. Accordingly it is well supposed to be prepared for the usage in the real time.

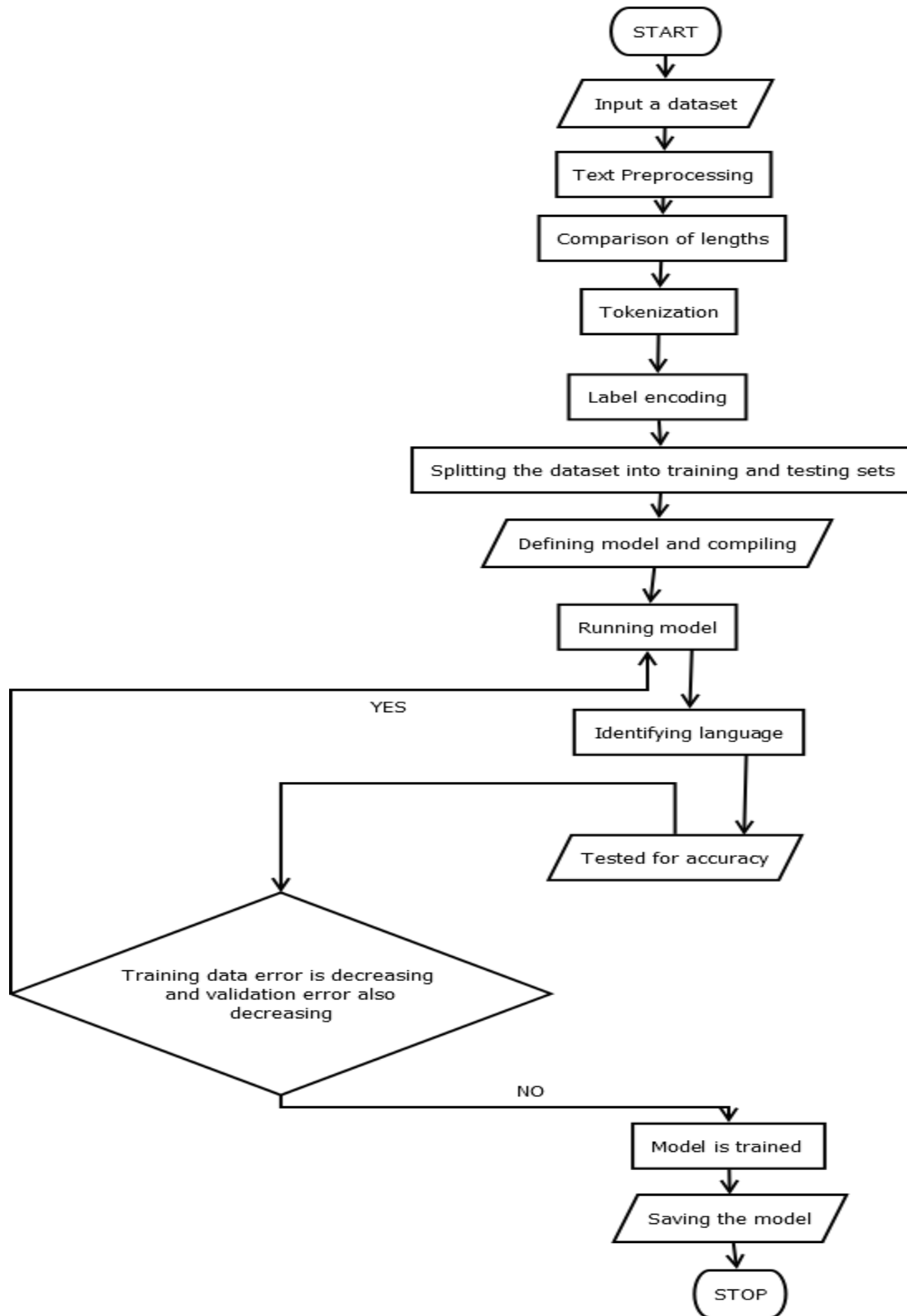


Diagram-3

System Implementation from Developer's perspective

5. Methodology

5.1 Platforms/IDE's:

Anaconda-Anaconda is essentially a Python platform that empowers the utilization of several IDEs. Anaconda isn't restricted to Python any longer and furthermore accompanies interfaces and conditions for R language. It empowers the implicit establishment of programming language platforms for Python and R as well as IDEs like PyCharm, Spyder, or Jupyter Notebook.

Jupyter-The Jupyter Notebook is an IDE that grants you to make and share records that contain live code, depictions and text. Utilities of such an IDE are information cleaning and transformation, real time depictions, data insight, AI and many more. Jupyter Notebooks are an open file system reliant upon JSON.

Spyder- Spyder is a free and open-source logical IDE written in Python, for Python, and planned by and for researchers, specialists and information experts. It includes a remarkable mix of the high level altering, investigation, troubleshooting, and profiling of a complete aspect of that of a tool used in software development with the information investigation, intuitive execution, profound assessment, and very good representation capacities.

5.2 Libraries:

- Scikit-learn, Keras, TensorFlow — has text processing capabilities.
- NLTK — Natural language toolkit.
- SpaCy — is an industrial strength NLP package with many practical tools in a nice API.
- Other libraries — TextBlob, gensim, Stanford CoreNLP, OpenNLP.

5.3 NLP method:

1) Removal of punctuation-

“Who is that person?” = “Who is that person”

2) Removal of stop words-

“This is introduction to NLP” = “introduction NLP”

3) Stemming-Cuts the inflected words to their root form.

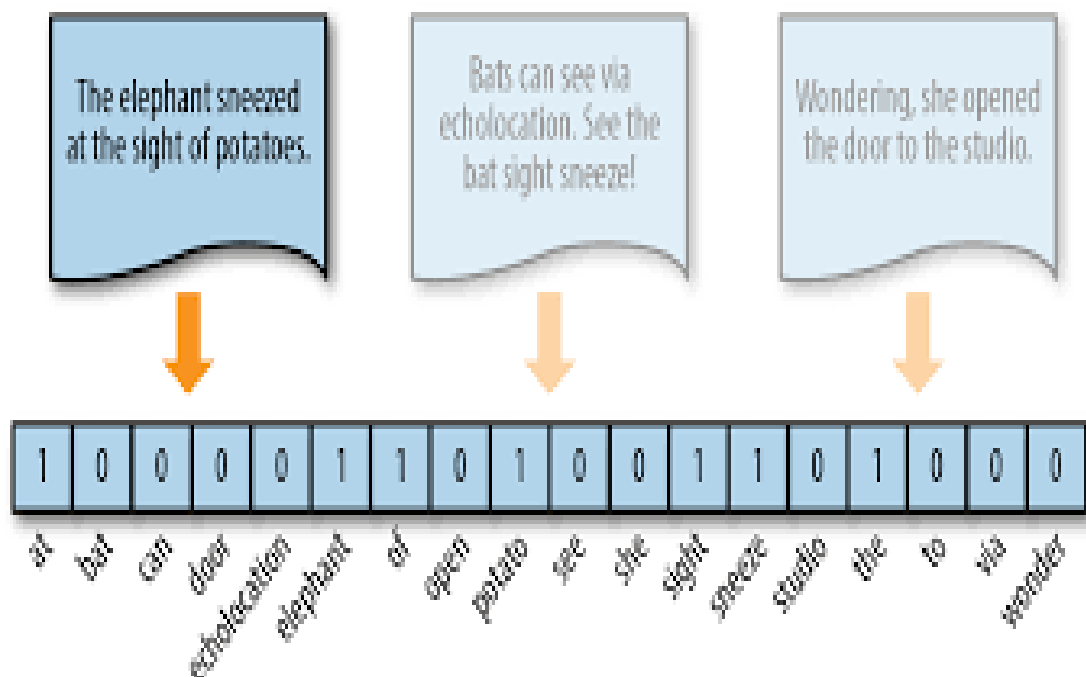
“I like travelling” = “I like travel”

4) Lemmatization-reduces various inflected forms of a word into a single form.

“There are many colours in poster” = “There are many colour in poster”

5) Vectorization (bag of words problem)-

For Natural Language Processing (NLP) to work, it always requires to transform natural language (text and audio) into numerical form that is in terms of frequency.



Vectorization in Natural Language Processing

6.Implementation

6.1 Datasets:

The datasets for the project have been taken from the Tatoeba Project which is a language project website where there are different datasets available for the purpose of the NLP projects.

We have taken the datasets for the french and german language which will have an english sentence delimited by a tab following a german translated sentence and then followed by the attributes of the owner which will be eliminated in the data preprocessing phase.

```
1 Go. Geh. — CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8597805 (Roujin)
2 Hi. Hallo! CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #380701 (cбургmer)
3 Hi. Grüß Gott! CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #659813 (Esperantostern)
4 Run! — Lauf! — CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #941078 (Fingerhut)
5 Run. — Lauf! — CC-BY 2.0 (France) Attribution: tatoeba.org #4008918 (JSakuragi) & #941078 (Fingerhut)
6 Wow! — Potzdonner! CC-BY 2.0 (France) Attribution: tatoeba.org #52027 (Zifre) & #2122382 (Pfirsichbaeumchen)
7 Wow! — Donnerwetter! — CC-BY 2.0 (France) Attribution: tatoeba.org #52027 (Zifre) & #2122391
  (Pfirsichbaeumchen)
8 Fire! — Feuer! CC-BY 2.0 (France) Attribution: tatoeba.org #1829639 (Spamster) & #1958697 (Tamy)
9 Help! — Hilfe! CC-BY 2.0 (France) Attribution: tatoeba.org #435084 (lukaszpp) & #575889 (MUIRIEL)
```

The structure of German Dataset

```
1 Go. Va ! — CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev)
2 Hi. Salut ! CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji)
3 Hi. Salut. CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #4320462 (gillux)
4 Run! — Cours ! CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #906331 (sacredceltic)
5 Run! — Courez ! — CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #906332 (sacredceltic)
6 Who? — Qui ? — CC-BY 2.0 (France) Attribution: tatoeba.org #2083030 (CK) & #4366796 (gillux)
7 Wow! — Ça alors ! CC-BY 2.0 (France) Attribution: tatoeba.org #52027 (Zifre) & #374631 (zmoo)
8 Fire! — Au feu ! — CC-BY 2.0 (France) Attribution: tatoeba.org #1829639 (Spamster) & #4627939 (sacredceltic)
9 Help! — À l'aide ! CC-BY 2.0 (France) Attribution: tatoeba.org #435084 (lukaszpp) & #128430 (sysko)
```

The structure of the French Dataset

6.2 Language Translation:

The following steps have been followed in implementing the translation system.

1. Importing the files and libraries.

```
import string
import re
from numpy import array, argmax, random, take
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding, RepeatVector
from keras.preprocessing.text import Tokenizer
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.sequence import pad_sequences
from keras.models import load_model
from keras import optimizers
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_colwidth', 200)
```

2. Reading of the text from the file.

```
filename = 'deu.txt'
def read_text(filename):
    # open the file
    file = open(filename, mode='rt', encoding='utf-8')

    # read all text
    text = file.read()
    file.close()
    return text

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


3. Text pre-processing:
 - i. Removal of spaces and tabs.
 - ii. Removal of punctuations.
 - iii. Conversion to lowercase.

```
data = read_text("deu.txt")
deu_eng = to_lines(data)
deu_eng = array(deu_eng)

deu_eng = deu_eng[:50000,:]

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

for i in range(len(deu_eng)):
    deu_eng[i,0] = deu_eng[i,0].lower()
    deu_eng[i,1] = deu_eng[i,1].lower()
```

4. Comparison of the text lengths.

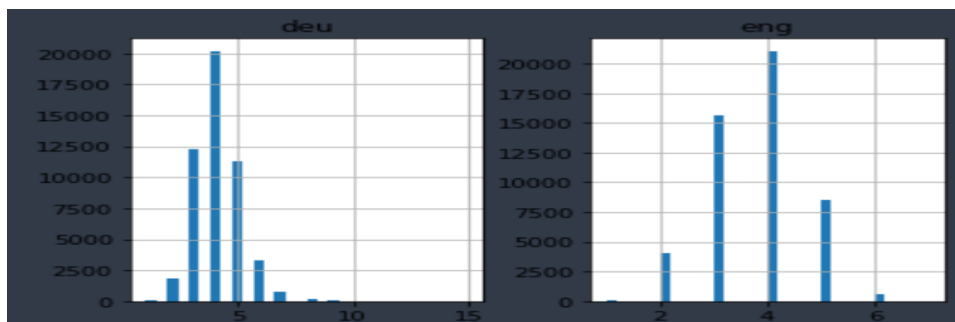
```
eng_l = []
deu_l = []

# populate the lists with sentence lengths
for i in deu_eng[:,0]:
    eng_l.append(len(i.split()))

for i in deu_eng[:,1]:
    deu_l.append(len(i.split()))

length_df = pd.DataFrame({'eng':eng_l, 'deu':deu_l})

length_df.hist(bins = 30)
plt.show()
```



5. Tokenization of lists.

```
def tokenization(lines):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer

eng_tokenizer = tokenization(deu_eng[:, 0])
eng_vocab_size = len(eng_tokenizer.word_index) + 1

eng_length = 8
print('English Vocabulary Size: %d' % eng_vocab_size)

English Vocabulary Size: 6241

deu_tokenizer = tokenization(deu_eng[:, 1])
deu_vocab_size = len(deu_tokenizer.word_index) + 1

deu_length = 8
print('Deutsch Vocabulary Size: %d' % deu_vocab_size)

Deutsch Vocabulary Size: 10326
```

6. Encoding of lists and Splitting of data into training and testing models.

```
def encode_sequences(tokenizer, length, lines):
    # integer encode sequences
    seq = tokenizer.texts_to_sequences(lines)
    # pad sequences with 0 values
    seq = pad_sequences(seq, maxlen=length, padding='post')
    return seq

from sklearn.model_selection import train_test_split

# split data into train and test set
train, test = train_test_split(deu_eng, test_size=0.2, random_state = 12)

trainX = encode_sequences(deu_tokenizer, deu_length, train[:, 1])
trainY = encode_sequences(eng_tokenizer, eng_length, train[:, 0])

# prepare validation data
testX = encode_sequences(deu_tokenizer, deu_length, test[:, 1])
testY = encode_sequences(eng_tokenizer, eng_length, test[:, 0])
```

7. Defining the model with respect to the dense layers, activation functions and other parameters.

```
def define_model(in_vocab,out_vocab, in_timesteps,out_timesteps,units):
    model = Sequential()
    model.add(Embedding(in_vocab, units, input_length=in_timesteps, mask_zero=True))
    model.add(LSTM(units))
    model.add(RepeatVector(out_timesteps))
    model.add(LSTM(units, return_sequences=True))
    model.add(Dense(out_vocab, activation='softmax'))
    return model

model = define_model(deu_vocab_size, eng_vocab_size, deu_length, eng_length, 512)

rms = optimizers.RMSprop(lr=0.001)
model.compile(optimizer=rms, loss='sparse_categorical_crossentropy')
```

8. Defining the optimizers and compiling the model, running the epochs and saving the model.

```
filename = 'model.h1'
checkpoint = ModelCheckpoint(filename, monitor='val_loss', verbose=1, save_best_only=True, mode='min')

# train model
history = model.fit(trainX, trainY.reshape(trainY.shape[0], trainY.shape[1], 1),
                    epochs=30, batch_size=512, validation_split = 0.2,callbacks=[checkpoint],
                    verbose=1)
```

6.3 Language Identification:

1. Importing libraries and reading the data from the CSV file.

```
import nltk
import tensorflow as tf
import keras
import seaborn as sns
from nltk.corpus import stopwords
print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'the
in', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'wer
e', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
use', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'afte
r', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not',
'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'l
l', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'has
n', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 's
houldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

df = pd.read_csv("dataset.csv")
```

2. Then pre-processing the data i.e.,
 - i. Checking for the number of objects in the dataset.
 - ii. Converting the text to lowercase.

```
df["Text"] = df["Text"].str.lower()
```

- iii. Getting basic info.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22000 entries, 0 to 21999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   Text        22000 non-null  object 
 1   language    22000 non-null  object 
dtypes: object(2)
memory usage: 343.9+ KB
```

3. Build a tokenizer to tokenize the data.

```
max_words = 50000
max_len = 100

tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words = max_words)
tokenizer.fit_on_texts(list(df['Text']))
train_df = tokenizer.texts_to_sequences(list(df['Text']))

train_df = tf.keras.preprocessing.sequence.pad_sequences(train_df, maxlen = max_len)
```

4. Build a label encoder to label encode the languages column values.

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
le = preprocessing.LabelEncoder()

le.fit(Y)

Y2 = le.fit_transform(Y)

total_languages = df['language'].nunique()

Y2 = keras.utils.to_categorical(Y2, num_classes=total_languages)
```

5. Split the data 4:1 and build a keras model using the sequential embedding with softmax activation function.

```
X_train,X_test,Y_train,Y_test = train_test_split(train_df,Y2)

embedding_dims = 500
vocab_size = len(tokenizer.word_index)+1

model = tf.keras.Sequential([tf.keras.layers.Embedding(input_dim = vocab_size,output_dim = embedding_dims,i
                        tf.keras.layers.Flatten(),
                        tf.keras.layers.Dense(total_languages,activation=tf.nn.softmax)])
```

```
model.summary()
```

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|-----------------------|------------------|-----------|
| embedding (Embedding) | (None, 100, 500) | 136984000 |
| flatten (Flatten) | (None, 50000) | 0 |
| dense (Dense) | (None, 22) | 1100022 |

=====
Total params: 138,084,022
Trainable params: 138,084,022
Non-trainable params: 0

7. Compile the model using categorical cross entropy loss function and then fit and run the model to a maximum of 3 epochs.

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
model.fit(np.array(X_train), np.array(Y_train), epochs=3)
```

```
Epoch 1/3
516/516 [=====] - 997s 2s/step - loss: 0.6249 - accuracy: 0.8321
Epoch 2/3
516/516 [=====] - 1006s 2s/step - loss: 0.1120 - accuracy: 0.9478
Epoch 3/3
516/516 [=====] - 997s 2s/step - loss: 0.0865 - accuracy: 0.9538
```

```
<tensorflow.python.keras.callbacks.History at 0x2363530d088>
```

6.4 Integration of both:

We integrate both the models by saving them individually and calling them together in a separate code file. Then we try to use both together where we create a function that takes the parameters that is the sentence which we want to identify and translate.

At first the sentence is identified and then translated using both models then the function shall return both the values as a list of strings.

This will be taken by another function that handles the HTML forms and the results shall be displayed onto the HTML page.

```
9 model = load_model('translator_mod/model.h1.24_jan_19')
10 def detect_and_translate(text,target_lang):
11     result_lang = detect(text)
12     if result_lang == target_lang:
13         return [text, result_lang]
14     else:
15         result = translator_model(text,result_lang)
16         return [result, result_lang]
```

The FLASK Code part-1

```
def predict():
    if request.method == 'POST':
        message = request.form['message']
        lis = detect_and_translate(message,target_lang='en')
        my_prediction = str(lis[0])
        result_lan = str(lis[1])
        if result_lan == "de":
            result_lan = "(Language detected was German)"
        elif result_lan == "fr":
            result_lan = "(Language detected was French)"
        else:
            result_lan = "(Langauge was neither french nor german)"

        my_pred = my_prediction + result_lan
        return render_template('result.html',prediction = my_pred)
```

The FLASK Code part-2

6.5 Front end:

The HTML and CSS file are written and saved in the templates and static folders respectively. There are two HTML files developed, one for the page where we enter a sentence to be translated and the other is for the page where results are displayed.

Thus the Flask API has been used alongside python so as to handle the forms in HTML. Thus we have used .py extension to save the frontend file in python flask application.

Then we have also used the Anaconda cmd prompt to launch our flask application on the Chrome web browser.



```
home.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
  <title>Home</title>
  <!-- <link rel="stylesheet" type="text/css" href="../static/css/styles.css"> -->
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>

  <header>
    <div class="container">
      <div id="brandname">
        LANGUAGE IDENTIFICATION AND TRANSLATION
      </div>
      <h2>Project</h2>
    </div>
  </header>

  <div class="ml-container">

    <form action="{{ url_for('predict')}}" method="POST">
      <p>Enter your text to be translated here</p>
      <!-- <input type="text" name="comment"/> -->
      <textarea name="message" rows="4" cols="50"></textarea>
      <br/>
    </form>
```

The Home Page Code

```
result.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <link rel="stylesheet" type="text/css" href="{ url_for('static', filename='css/styles.css') }}">
</head>
<body>

  <header>
    <div class="container">
      <div id="brandname">
        LANGUAGE IDENTIFICATION AND TRANSLATION
      </div>
      <h2>Project</h2>
    </div>
  </header>
  <p style="color:blue;font-size:20;text-align: left;"><b>Translated sentence and identified language is: </b></p>
  <div class="results">
    <h3>{{prediction}}</h3>
  </div>

</body>
</html>
```

The Result Page Code

```
body{
  font:15px/1.5 Arial, Helvetica,sans-serif;
  padding: 0px;
  background-color:#f4f3f3;
}

.container{
  width:100%;
  margin: auto;
  overflow: hidden;
}

header{
  background:#03A9F4; #354348;
  border-bottom:#448AFF 3px solid;
  height:120px;
  width:100%;
  padding-top:30px;
```

The CSS Code Part-1


```

21
22 .main-header{
23     text-align:center;
24     background-color: blue;
25     height:100px;
26     width:100%;
27     margin:0px;
28 }
29 #brandname{
30     float:left;
31     font-size:30px;
32     color: #fff;
33     margin: 10px;
34 }
35
36 header h2{
37     text-align:center;
38     color:#fff;
39
40 }

```

The CSS Code Part-2

```

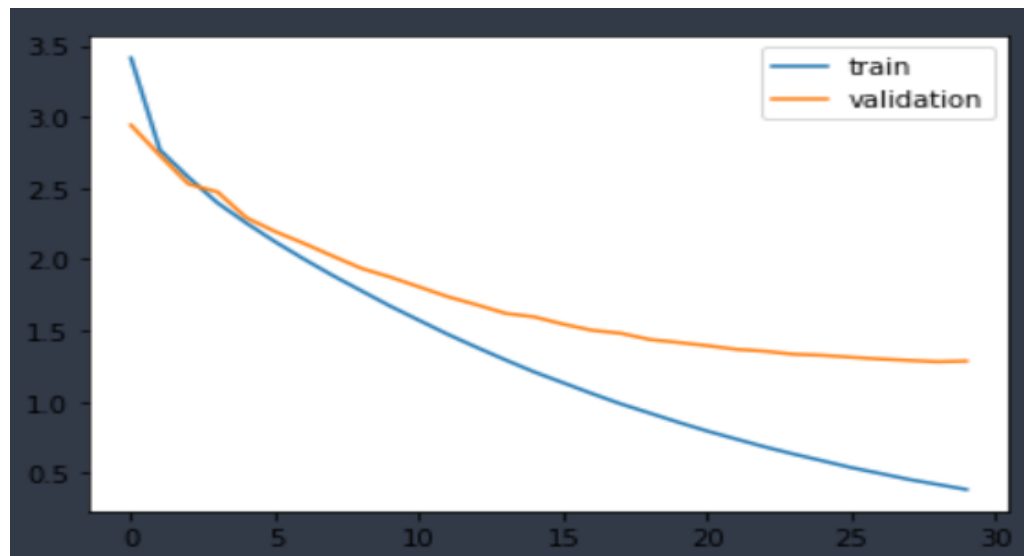
41
42
43
44 .btn-info {background-color: #2196F3;
45     height:40px;
46     width:100px;} /* Blue */
47 .btn-info:hover {background: #0b7dda;}
48
49
50 .resultss{
51     border-radius: 15px 50px;
52     background: #345fe4;
53     padding: 20px;
54     width: 200px;
55     height: 150px;
56 }

```

The CSS Code Part-3

7. System Testing

7.1 Language Translation Testing:



Results showing the optimum no. of epochs

As one could observe the model has stabilised after 22 epochs or so. Thus any further training would lead to overfitting.

The results which we got were quite intuitive. The results indicate that the model which we have implemented is able to grasp the context of the sentence and translate the same. Thus the model we implemented using LSTM with encoder and decoder methodology has worked out and is now ready for the next phase that is the language identification systems.

```
text = ["Une fois que vous connaissez tous les éléments, il n'est pas difficile de rassembler une phrase."]

test_text = tokenizer.texts_to_sequences(text)
test_text = tf.keras.preprocessing.sequence.pad_sequences(test_text, maxlen=max_len)
predictions = model.predict(test_text)
out = predictions.argmax()
print(1e.inverse_transform([out]))
print(predictions)

['French']
[[2.0560618e-03 2.3356976e-03 1.4917282e-02 8.7683310e-04 1.0380887e-02
 8.9271063e-01 6.5082818e-04 1.2924318e-03 9.2817293e-03 3.6108880e-03
 7.5576487e-03 9.9017855e-04 7.9061072e-03 1.7024969e-03 9.8757949e-03
 3.0762965e-03 1.8487133e-02 1.0203347e-03 2.5317175e-03 4.9194507e-03
 2.6453331e-03 1.1741561e-03]]
```

The Result of the Identification Model

Then we tend to pick up the labels encoded for each language and then we try to give them a trial test over a sentence from a language. At first we gave french and then we gave another french sentence. Even though the model has been trained on the german and french, the 2nd french sentence was quite misrepresented by the model built.

7.2 Language Identification Testing:

As one could observe the number of epochs taken is 3. We have evaluated the model on the testing data where we got an accuracy of 93%.

```
model.evaluate(np.array(X_test),np.array(Y_test))
```

```
172/172 [=====] - 1s 5ms/step - loss: 0.1764 - accuracy: 0.9304
```

```
[0.17641477286815643, 0.930363655090332]
```

Accuracy of the Identification Model

| | | |
|------|-----------------------|-----------------------|
| 4490 | tom seemed sleepy | tom seemed tired |
| 157 | i didnt strangle tom | i didnt hated tom |
| 4338 | i know that voice | i know that guy |
| 4139 | let me have a look | let see |
| 4950 | he was shy at first | he was born poor dogs |
| 4400 | i love horses | i love spinach |
| 629 | i want toms money | i want the money |
| 3955 | wear warm clothes | turn on the robe |
| 2136 | tom was objective | tom was confused |
| 519 | he offered to help me | he told me to |
| 7521 | wood burns easily | italy is easy |
| 8352 | he has small feet | hes has small feet |
| 9010 | look at this photo | look at the picture |
| 6557 | i live in boston | i live in boston |
| 2739 | she walks | she started |

The Results of the Translation Model

8. Results

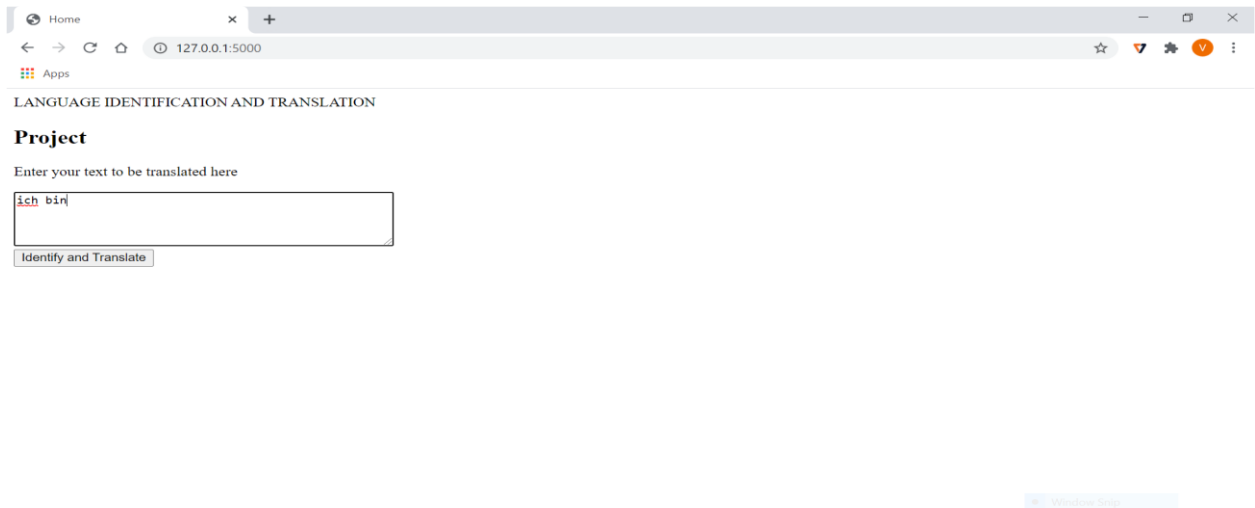
The following are the snapshots from the results that we got by running the flask code. At first we gave the input as ich bin in German and then pressed the button below. On the 2nd page we would get the results as I am (Language detected was German) which is the accurate output.

```
Anaconda Prompt (Anaconda3) - python frontendcd.py
(base) C:\Users\anand>cd desktop

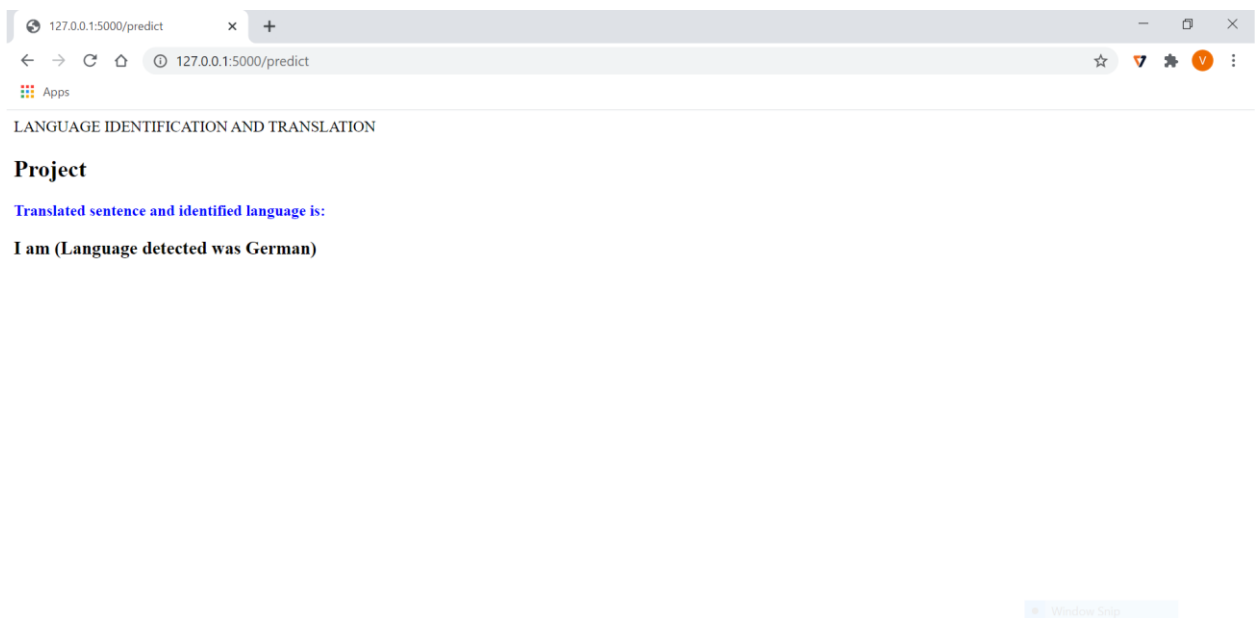
(base) C:\Users\anand\Desktop>cd miniproject

(base) C:\Users\anand\Desktop\MiniProject>python frontendcd.py
C:\Users\anand\Anaconda3\lib\site-packages\sklearn\externals\joblib\_init__.py:15: FutureWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with skikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
* Serving Flask app "frontendcd" (lazy loading)
* Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
C:\Users\anand\Anaconda3\lib\site-packages\sklearn\externals\joblib\_init__.py:15: FutureWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with skikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
* Debugger is active!
* Debugger PIN: 140-817-970
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [18/Apr/2021 16:22:00] "B[37mGET / HTTP/1.1B[0m" 200 -
127.0.0.1 - - [18/Apr/2021 16:22:00] "B[33mGET /static/css/styles.css HTTP/1.1B[0m" 404 -
127.0.0.1 - - [18/Apr/2021 16:22:00] "B[33mGET /favicon.ico HTTP/1.1B[0m" 404 -
127.0.0.1 - - [18/Apr/2021 16:22:41] "B[37mPOST /predict HTTP/1.1B[0m" 200 -
127.0.0.1 - - [18/Apr/2021 16:22:41] "B[33mGET /static/css/styles.css HTTP/1.1B[0m" 404 -
```

The Running of the Code



The Home Page



The Result Page

9. Conclusion:

LSTM with encoder and decoder models has resulted in a greater improvement in terms of accuracy and could be used to build a LID that is a Language IDentification and translation model where the inputs are in the format of text. The features extraction capabilities during the preprocessing phase could be utilized to extract the data about the features of the text data and the same could be used in classifying or identifying the language by using the LSTM powerful information extraction that is helpful in classification based on that information. The problem of language identification in the domain of images and the translation of the extracted language input into a text format could be achieved using this LSTM and encoder and decoder capabilities.

There are restrictions to our task like the media information info couldn't be given, for example, the sound or video input documents, etc. This is on the off chance that the RNN with LSTM which we could utilize couldn't deal with the tremendous measure of information that is produced by the mixed media input records. To address this issue the idea of GRU's that is Gated Recurrent Units may be utilized as they could deal with tremendous volumes of I - vectors produced by such inputs. This is a technique proposed in the research paper of Language Identification Using Deep Convolutional Recurrent Neural Networks by Christian Bartz, Tom Herold , Haojin Yang and Christoph Meinel.

References:

- Automatic Language Identification Using Deep Neural Networks by Ignacio Lopez-Moreno , Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez , Pedro Moreno.
- Language Identification from Text Documents by Priyank Mathur, Arkajyoti Misra, Emrah Budur.
- Language Identification Using Deep Convolutional Recurrent Neural Networks by Christian Bartz, Tom Herold , Haojin Yang and Christoph Meinel.
- Deep Learning A-Z course by Kirill Eremenko and Ponteves from Udemy.
- Deep Learning course from Analytics Vidya.
- Machine Learning A-Z course by Kirill Eremenko and Ponteves from Udemy.
- Data Science Today website for RNN and CNN.
- Y. K. Muthusamy, E. Barnard and R. A. Cole, "Reviewing automatic language identification," in *IEEE Signal Processing Magazine*, vol. 11, no. 4, pp. 33-41, Oct. 1994, doi: 10.1109/79.317925.
- E. Ambikairajah, H. Li, L. Wang, B. Yin and V. Sethu, "Language Identification: A Tutorial," in *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 82-108, Second quarter 2011, doi: 10.1109/MCAS.2011.941081.
- M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," in *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31-, Jan. 1996, doi: 10.1109/TSA.1996.481450.
- L. Ferrer, N. Scheffer and E. Shriberg, "A comparison of approaches for modeling prosodic features in speaker recognition," *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 4414-4417, doi: 10.1109/ICASSP.2010.5495632.