

# Sign Language Translation

Project guide :  
Mr. Deepak Kumar Gaur

BY:

Aman Agrawal - EN18CS303004  
Arran P Gonsalves - EN18CS303008  
Kevin Gajjar - EN18CS303018

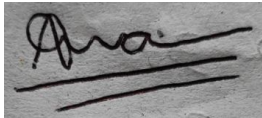
# Declaration

We hereby declare that the project entitled “**Sign Language Translation**” submitted in partial fulfillment for the award of degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Mr. Deepak Kumar Gaur**, Faculty of Engineering, Medi-Caps University Indore is an authentic work.


Further, We declare that the content of this Project work, in full or in parts, have neither been taken from any other sources nor have been submitted to any other Institute or University for the award of any degree or diploma.

**BY:-**

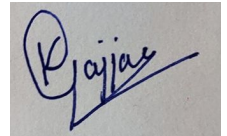
Aman Agrawal

A handwritten signature in black ink, appearing to read 'Aman', with two horizontal lines drawn below it.

Arran Pratik Gonsalves

A handwritten signature in blue ink, appearing to read 'Arran', with a large, stylized 'P' and 'G' below it.

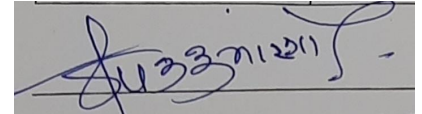
Kevin Gajjar

A handwritten signature in blue ink, appearing to read 'Kevin', with a large, stylized 'G' and 'J' below it.

# Certificate

I, **Mr. Deepak Kumar Gaur** certify that the project “**Sign Language Translation**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ by **Aman Agrawal, Arran P Gonsalves, Kevin Gajjar** is the record carried out by them under my guidance and that work has not formed the basis if any other degree elsewhere.

**Signature of Guide**

A handwritten signature in blue ink, appearing to read 'Deepak Kumar Gaur', is written over a horizontal line on a light gray background.

Mr. Deepak Kumar Gaur

# Introduction

Sign language is a visual means of communication through hand signals, gestures, facial expressions, and body language. It's the main form of communication for the Deaf and Hard-of-Hearing community, as well as other groups of people. There is no single sign language used around the world. It has developed naturally through different groups of people interacting with each other, so there are many varieties.

One of the major shortcomings of society is the social barrier that is created amongst disabled or handicapped persons. Communication, which is the basis of human progress, often tends to be an obstacle for those unfortunate. The root cause of this gap is that while deaf and mute persons use sign language to communicate among themselves, normal people are either reluctant to learn it or are unable to comprehend the same.



There exist different ways of communication such as impaired rely on lip reading, Sign Language Interpreter who translate sign language to speech, virtual assistance through different devices. But a lot of these have certain shortcomings like reliability, easy accessibility, cost and availability.

The aim is to ease the method for translation between Sign language and the English language which helps to overcome the shortcomings of preexisting methods and thus eliminate the barrier of communication. With the use of Machine Learning and Natural Language Processing, propose a method which helps to translate Sign-to-Text and Text-to-Sign language. Generating meaningful sentences after translation of sign language and using ASL grammar to derive Signs from words provided after tokenization of text entered by the user.



# Problem statement

Sign language is a language that uses manual communication methods such as facial expressions, hand gestures and bodily movements to convey information. There is a barrier of communication that exists between normal and impaired people in terms of understanding each other in terms of a universal language. Most are unaware of sign language and resources to learn or having a medium is not widely available. Due to this the impaired are unable to express themselves freely. The current translation of Sign Language does not include whole word or sentence translation which affects the speed of communication.

# Objective

The aim is to ease the translation of sign to English language also from English language to sign language. Few works have been done to generate a system that translates the whole sentence with the meaning from text-to-sign and have real time conversion of sign-to-text.

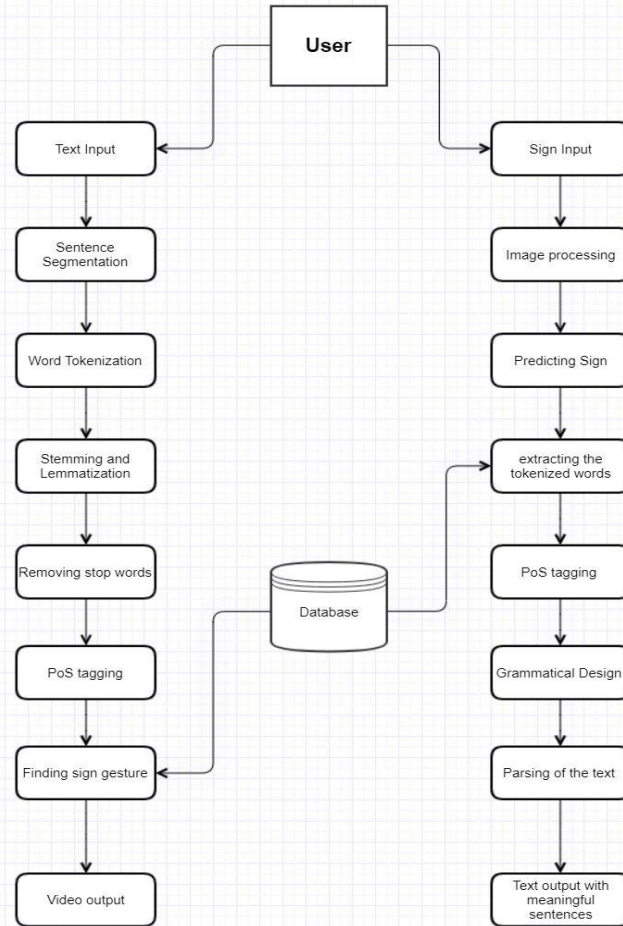
- For sign-to-text we will use Deep Learning to process the dataset and train a CNN model using Keras library to identify frames of the Sign Language. The motive is to develop a model that can be used on varying datasets (images or videos) which are properly preprocessed and extract the meaning of signs in text.
- Propose a method to make those groups of words into a meaningful sentence by utilizing parts of speech tagging(PoS), parsing and grammar.
- For text-to-sign translation our model uses Natural Language processing techniques which will process the text in a form which can directly give us the desired output that can be converted into sign.



# WorkFlow Structure



## Text to sign



## Sign to text

# Requirements

## User Requirements:

A text input which is to be converted into sign and sign language input from the person with disability.

## Software Requirements:

- Technology Used: Machine Learning, Deep Learning, Natural Language Processing.
- Tools: Google Colab
- Language: Python 3
- Libraries: NLTK, Spacy, Keras, Tensorflow, Pandas, Numpy

# Database

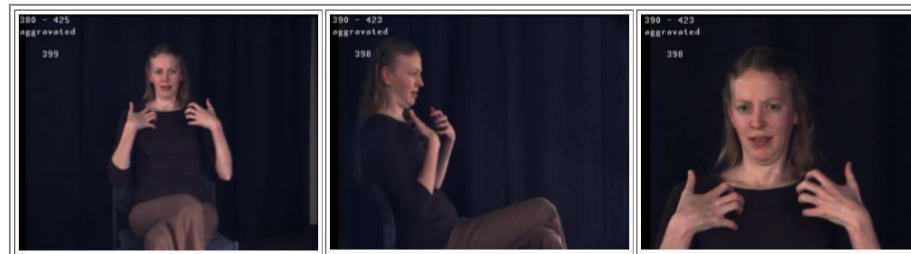
Data is fundamental in any research project, especially in developing a system for recognising and translating sign language. Since the grammar and the mode of communication differ so much it is important to have a large dataset that is well labeled and can provide lexical meaning for signs for improved translation.

We are using 2 datasets namely:

- American Sign Language Lexicon Video Dataset (ASLLVD) which consists of video data from multiple angle for words that are represented in sign language by a number of interpreters. Due to the terms of use we cannot download the dataset and have to rely on the website to provide us with the data.
- ASL fingerspelling dataset which consist of images of signs used in ASL to represent letters (A-Z) and numbers (0-9). We will use this dataset to develop a model for recognition of sign language.

# American Sign Language Lexicon Video Dataset (ASLLVD)

The American Sign Language Lexicon Video Dataset (ASLLVD) consists of videos of >3,300 ASL signs in citation form, each produced by 1-6 native ASL signers, for a total of almost 9,800 tokens. This dataset includes multiple synchronized videos showing the signing from different angles. Linguistic annotations include gloss labels, sign start and end time codes, start and end handshape labels for both hands, morphological and articulatory classifications of sign type.



Link: <http://dai.cs.rutgers.edu/dai/s/signbank>

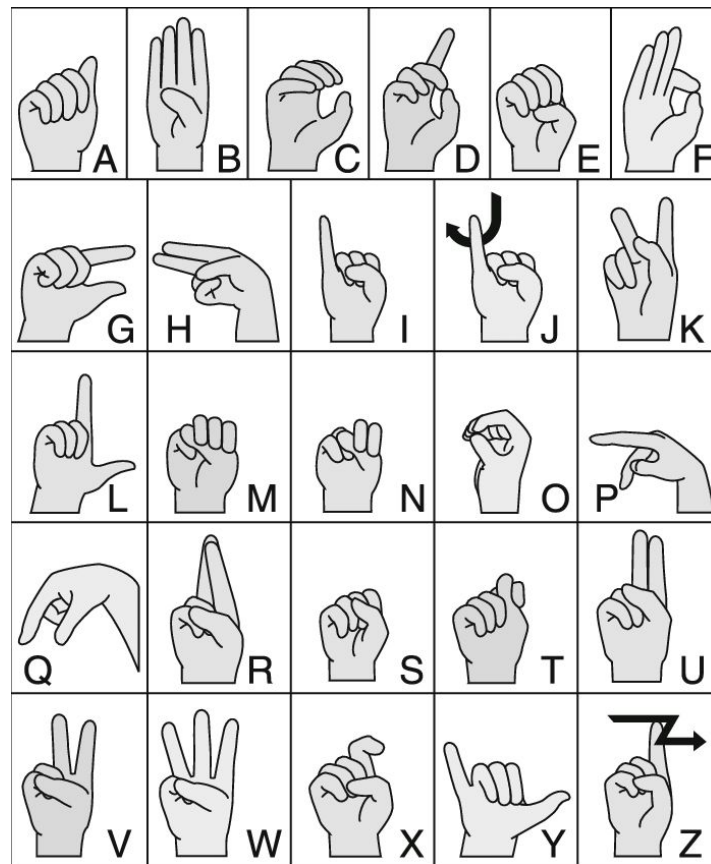
	Lexical Signs	Fingerspelled Signs	Name Signs	Loan Signs	Classifiers
One Handed	7469	946	499	564	88
One Handed - Marked # Hands	600	1	0	0	0
Two Handed	6556	0	21	9	266
Two Handed - Marked # Hands	173	0	0	8	1
Passive Base Arm	248	0	7	0	8

# ASL fingerspelling dataset

Fingerspelling is part of ASL and is used to spell out English words. In the fingerspelled alphabet, each letter corresponds to a distinct handshape. Fingerspelling is often used for proper names or to indicate the English word for something. It consists of 2515 processed photos showing all digits from 0 to 9 and all alphabet letters from a to z in ASL signs.

Link:

[https://drive.google.com/drive/folders/1YolcYyNMCKIi4yO0ginG\\_UnvC31wQKfx?usp=sharing](https://drive.google.com/drive/folders/1YolcYyNMCKIi4yO0ginG_UnvC31wQKfx?usp=sharing)



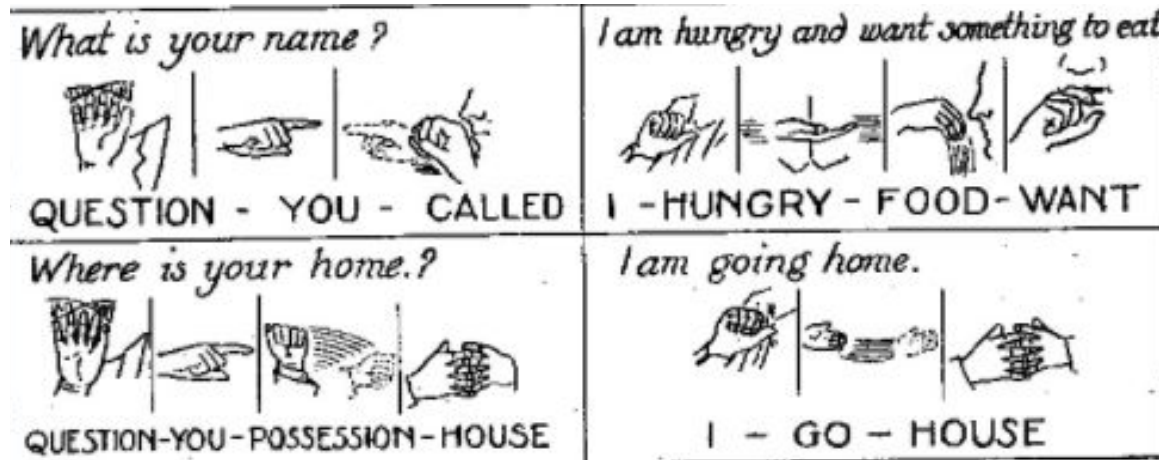


# Sign to Text Translation

# Phase 1 : Sign Recognition

In sign to text conversion we are developing a deep learning model for identification of american sign photos. We are using a dataset of ASL fingerspell available on kaggle which contains 2515 photos showing all digits from 0 to 9 and all alphabet letters from a to z.

By training the model with the respective dataset it will be able to recognize the image of a particular sign and give the output text which will be used to construct a word.



## Import the necessary libraries and dataset



```
1 #importing libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 import tensorflow as tf
8 from tensorflow import keras
```



```
1 #dataset
2 ds_asl_dir = "/tmp/asl_dataset/asl_dataset"
3 asl_ds = tf.keras.preprocessing.image_dataset_from_directory(ds_asl_dir)
```

Found 2515 files belonging to 36 classes.



## Data Preliminary Exploration

```
[ ] 1 pd.DataFrame(asl_ds.class_names)
```

### Image frames

#### Checking and displaying images with labels

```
✓ [43] 1 #Checking images and labels shapes (amount of images, height, width, color channels)
0s      2 for image_batch, labels_batch in asl_ds:
        3     print(image_batch.shape)
        4     print(labels_batch.shape)
        5     break
```

```
↳ (32, 256, 256, 3)
   (32,)
```

```
✓ [50] 1 #Displaying image samples
15s     2 plt.figure(figsize=(6, 6))
        3 for images, labels in asl_ds.take(1):
        4     for i in range(9):
        5         ax = plt.subplot(3, 3, i + 1)
        6         plt.imshow(images[i].numpy().astype("uint8"))
        7         plt.title(int(labels[i]))
        8         plt.axis("off")
```

✓  
25s

[50]



26



9



23



2



10



14



5



9



21



## Removing corrupt images



```
1 #Defining parameters for the loader
2
3 batch_size = 32
4 img_height = 64
5 img_width = 64
6
7 #Filtering out corrupted images
8
9 import os
10 num_skipped = 0
11 for folder_name in ("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"):
12     folder_path = os.path.join(ds_asl_dir, folder_name)
13     for fname in os.listdir(folder_path):
14         fpath = os.path.join(folder_path, fname)
15         try:
16             fobj = open(fpath, "rb")
17             is_jfif = tf.compat.as_bytes("JFIF") in fobj.peek(10)
18         finally:
19             fobj.close()
20         if not is_jfif:
21             num_skipped += 1
22             # Delete corrupted image
23             os.remove(fpath)
24 print("Deleted %d images" % num_skipped)
```

## Data Augmentation and Creating training and testing datasets

```
<>
26 #Augmenting the images
27
28 from keras.preprocessing.image import ImageDataGenerator
29 data_augmentation = ImageDataGenerator(rotation_range=15, rescale=1/255, zoom_range=0.1, horizontal_flip=True, width_shift_range=0.1, height_shift_range=0.1, validation_split=0.2)
30
31 #Setting train/test split
32
33 asl_train_ds = data_augmentation.flow_from_directory(directory=ds_asl_dir, target_size=(img_height, img_width), class_mode="categorical", batch_size=batch_size, subset="training")
34 asl_test_ds = data_augmentation.flow_from_directory(directory=ds_asl_dir, target_size=(img_height, img_width), class_mode="categorical", batch_size=batch_size, subset="validation")
```



```
Deleted 0 images
Found 2012 images belonging to 36 classes.
Found 503 images belonging to 36 classes.
```

## Data modelling

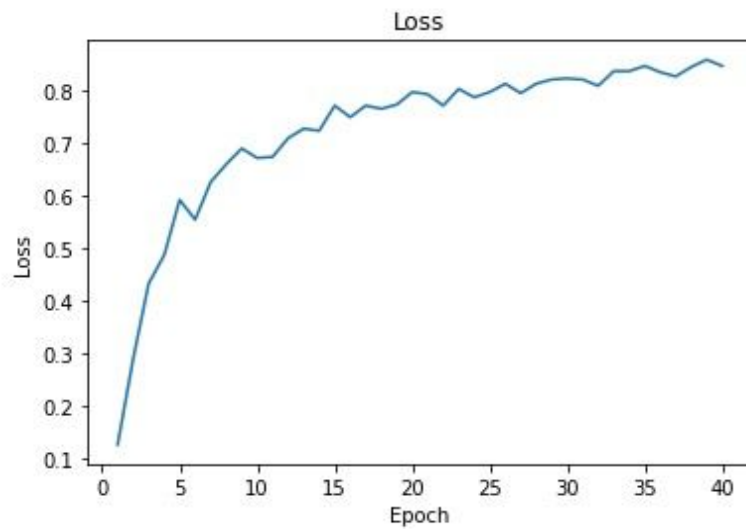
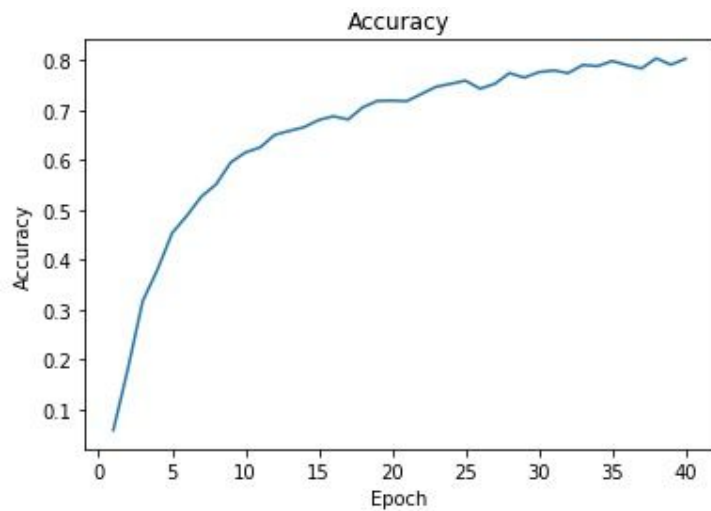
✓  
0s

```
[46] 1 from keras.layers import Conv2D,MaxPool2D,Dropout,Flatten,Dense
      2
      3 '''Checking if the data format i.e the RGB channel is coming first or last so,
      4 whatever it may be, model will check first and then input shape will be feeded accordingly.'''
      5
      6 from keras import backend as K
      7 if K.image_data_format() == "channels_first":
      8     input_shape = (3, img_height, img_width)
      9 else:
     10     input_shape = (img_height, img_width, 3)
     11
     12 #Creating a model
     13
     14 model_dl = keras.Sequential()
     15 model_dl.add(Conv2D(16,(3,3),activation="relu",input_shape=(input_shape)))
     16 model_dl.add(MaxPool2D(2,2))
     17 model_dl.add(Dropout(0.2))
     18 model_dl.add(Conv2D(32,(3,3),activation="relu"))
     19 model_dl.add(MaxPool2D(2,2))
     20 model_dl.add(Dropout(0.2))
     21 model_dl.add(Conv2D(64,(3,3),activation="relu"))
     22 model_dl.add(MaxPool2D(2,2))
     23 model_dl.add(Dropout(0.2))
     24 model_dl.add(Flatten())
     25 model_dl.add(Dense(128,activation="relu"))
     26 model_dl.add(Dropout(0.2))
     27 model_dl.add(Dense(36,activation="softmax"))
```

## Deep Learning Algorithm Implementation

```
[47] 1 #Compiling the neural network
      2
      3 model_dl.compile(optimizer="Adam", loss="categorical_crossentropy", metrics=["accuracy"])
      4
      5 #Fitting to the model
      6
      7 from keras.callbacks import EarlyStopping, ReduceLROnPlateau #Import callback functions
      8 earlystop=EarlyStopping(patience=10) #Monitor the performance. If it dips, then stop training
      9 learning_rate_reduce=ReduceLROnPlateau(monitor="val_acc", min_lr=0.001) #Change learning rate if not performing good enough
     10 callbacks=[earlystop, learning_rate_reduce]
     11
     12 model_dl.fit(asl_train_ds, validation_data=asl_test_ds, callbacks=callbacks, epochs=40)
```

## Accuracy and loss over time



## Saving the model to be used

```
✓ [48] 1 #Saving the model  
s      2  
      3 model_d1.save("model_d1.h5")  
      4  
      5 #Loading the model  
      6  
      7 model_d1 = keras.models.load_model("model_d1.h5") #look for local saved file
```



## Deployment of model

```
[49] 1 #We'll use any image sample from the Kaggle dataset to test it
      2
      3 from keras.preprocessing import image
      4
      5 #Creating a dictionary to map each of the indexes to the corresponding number or letter
      6
      7 dict = {0:"0",1:"1",2:"2",3:"3",4:"4",5:"5",6:"6",7:"7",8:"8",9:"9",10:"a",11:"b",12:"c",13:"d",14:"e",15:"f",16:"g",
      8         17:"h",18:"i",19:"j",20:"k",21:"l",22:"m",23:"n",24:"o",25:"p",26:"q",27:"r",28:"s",29:"t",30:"u",31:"v",32:"w",
      9         33:"x",34:"y",35:"z"}
      10
      11 #Predicting images
      12
      13 img = image.load_img("/hand1_g_left_seg_4_cropped.jpeg", target_size=(img_width, img_height))
      14 x = image.img_to_array(img)
      15 x = np.expand_dims(x, axis=0)
      16
      17 image = np.vstack([x])
      18 classes = np.argmax(model_dl.predict(x), axis=-1)
      19 probabilities = model_dl.predict(image, batch_size=batch_size)
      20 probabilities_formatted = list(map("{:.2f}%".format, probabilities[0]*100))
      21
      22 print(classes) #displaying matrix prediction position
      23
      24 print(f'The predicted image corresponds to "{dict[classes.item()]}"') #displaying matrix prediction position name (number or letter)
```

[16]

The predicted image corresponds to "g"

Image provided to be  
predicted


```
[16]  
The predicted image corresponds to "g"
```





## Phase 2 : Sentence formation

In this phase the formation of meaningful sentences using NLP techniques will take place. The group of words obtained from the above recognition technique will be used to generate a sentence that is grammatically correct. The sentences are formed using various POS tagging, Parsing and grammar.



**POS Tagging:** The words obtain from phase 1 passed to NLP The tag may indicate one of the parts-of-speech tag like noun (N), preposition (Prep), pronoun, adjective (Adj), verb (V), adverb (Adv), conjunction, and interjection . The input is tokenized and a training dataset will be used for detecting the equivalent part of speech of each word in the sentence. WordNet POS tagging Algorithm:

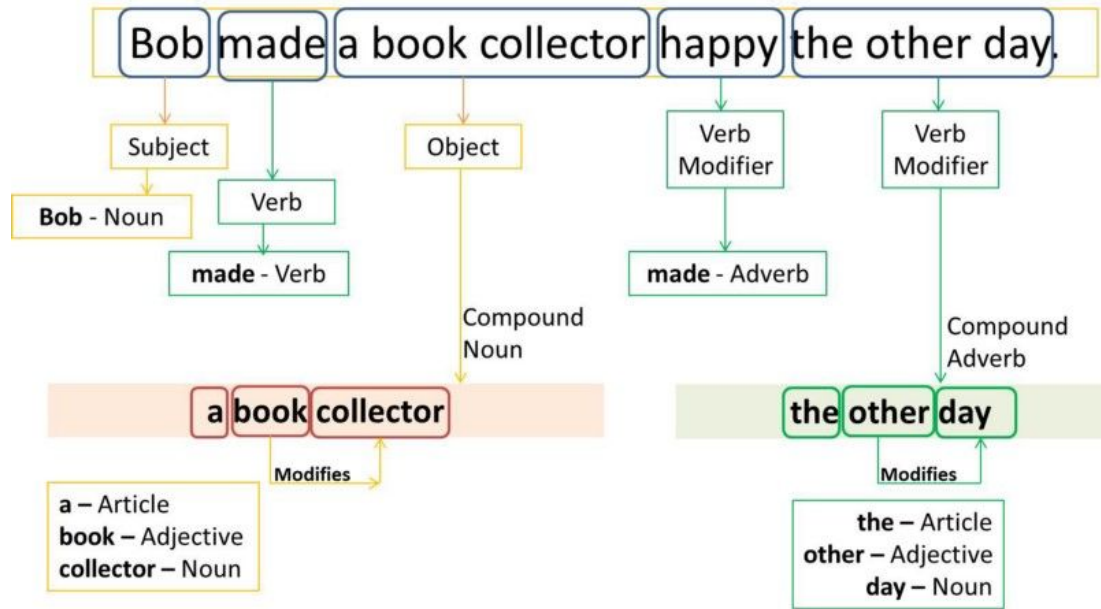
Take  $M$  as input and a specified list of POS tags such as N,V, Adj, Adv, Prep.

Search the words for specified tags in the training corpus.

Find the probability of tags for words in given corpus as follows:

$$P(t_i | w) = c(w, t_i) / (c(w, t_1) + \dots + c(w, t_k))$$

Where  $w$  = input word  $M$ ,  $t_i$  = tag for input word  $M$ ,  $c(w, t_i)$  is the number of times appear in corpus.



## Grammar Mapping:

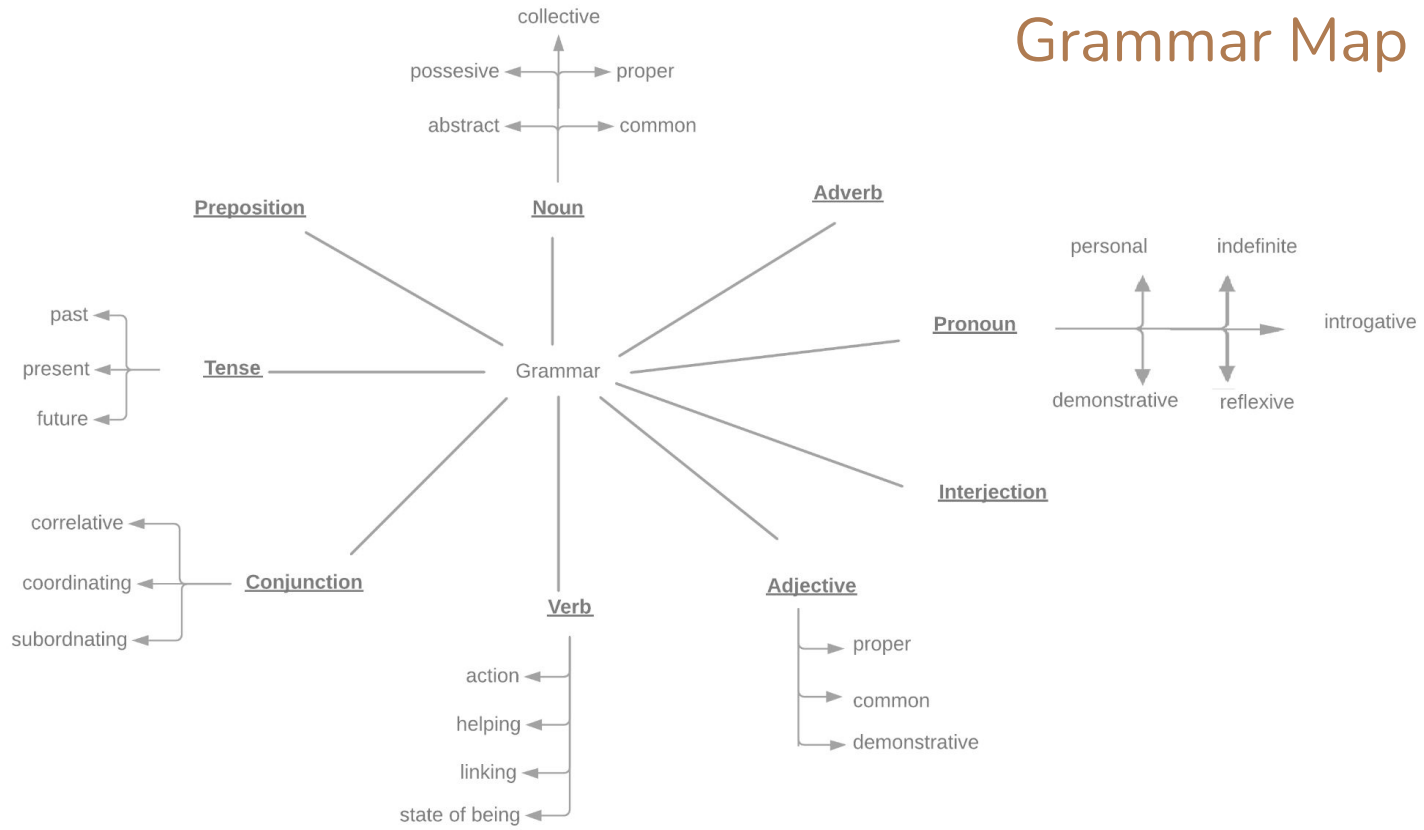
Grammar specifies the set of rules to define legal sentences in a particular language. These rules work together for constructing valid sentences. Since the grammar rules are different in ASL and English we must translate the grammar to obtain a significant sentence. The Simplest and easiest form of grammar is context free grammar. It is easy to deal with and write context-free grammar for people. Most of the time grammar is compromised by writing rules in context free style. These rules will be mapped so that we can use it for annotation of how the sentence will be formed.

It is useful to examine the use in following ways:

- Conceptual tool- It is used for ending the complexity of natural language by capturing and reporting.
- Formal notation – It is used to trace syntactical and semantical feature of language which is used by parser. Representation of Grammar design  $G = (V, T, P, S)$

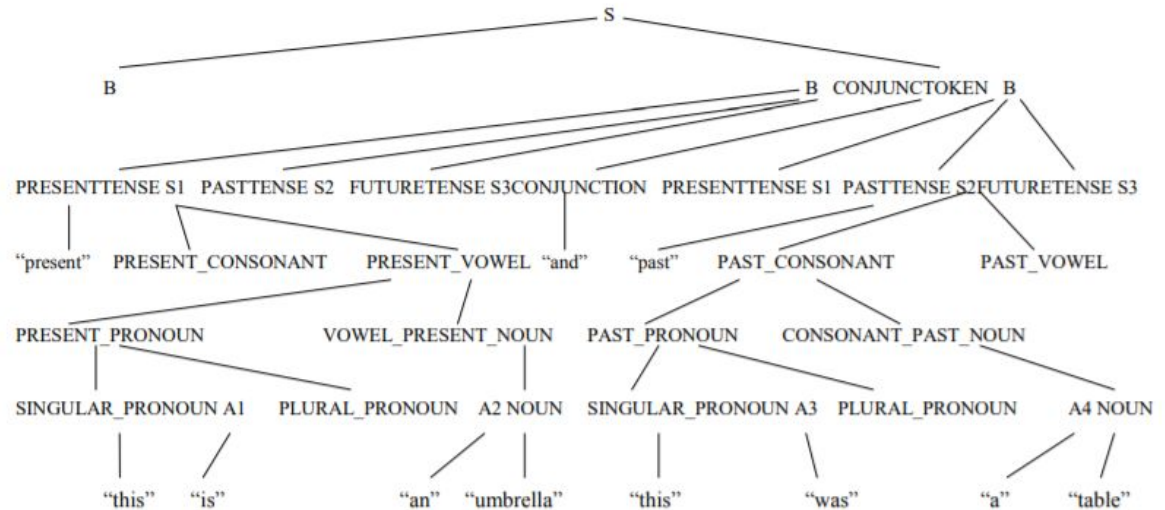
Where,  $G$  is the grammar which consists of a set of the production rule,  $T$  is the final set of a terminal symbol,  $V$  is the final set of a non-terminal symbol,  $P$  is a set of production rules,  $S$  is the start symbol which is used to derive the string.

# Grammar Map



**Parsing:** The Mapped rules of grammar will be used to parse the matrix of tagged words such that it will be processed with correct phrasing, tensing, articles and coherency. Using LALR parser to construct the parse tree. It is done in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). The parser starts with the input words, and tries to build trees from the words in upward direction, by applying rules from the grammar one at a time.

The parse is successful if the parser succeeds in structuring with the tree which is rooted in the start symbol (S) and covers all of the input. The parser begins by looking up the tags of each word and building a tree with the part of speech for each word. LALR parser, which is a simplified version of LR parser, is used to separate and analyze the text according to their production rules specified by formal grammar.





# Text to Sign Translation



# Phase 3: Text to sign

In Text-to-Sign we will be processing the text to generate tokens using Natural Language Processing methods and extracting a matrix which consists of a group of words whose signs are extracted from the dataset and can be understood in a meaningful way by a impaired person.

## Importing Libraries and fetching NLP documents

- **NumPy** is a Python library used for working with arrays. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. So here we have used numpy to process any array object.
- **Pandas** is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.
- **spaCy** is a free and open-source library for Natural Language Processing (NLP) in Python with a lot of in-built capabilities. It's becoming increasingly popular for processing and analyzing data in NLP.
- The **Natural Language Toolkit (NLTK)** is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets.

## Importing libraires

✓  
1s

```
[1] import pandas as pd
import numpy as np
import spacy
import nltk
```

✓  
0s

```
[2] nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]   /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]   date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

## Taking the text input from the user

### Text Input and processing

✓  
4s



```
input_text=input("enter the sentences seperated by full stop\n")
```



```
enter the sentences seperated by full stop  
What is your name? I am going home. i played with the dog.
```

✓  
0s

```
[4] input_text.lower()
```

```
'what is your name? i am going home. i played with the dog.'
```

## Defining Stop words

Stop words are a set of commonly used words in a language. Here they are words that do not exist in ASL grammar and cannot be represented in sign language.

Words included are:

'a','was','until','this', 'those', 'through','to', 'too','the', 'their', 'theirs','so', 'some',  
'such','or','have','for', 'from','can','as', 'at', 'be','am', 'an'

### Defining Stop Words

```
[6] stop_words=['a','was','until','this', 'those', 'through','to', 'too','the', 'their', 'theirs','so', 'some', 'such','or','have','for', 'from','can','as', 'at', 'be','am', 'an']
```

## Segmenting the sentence and tokenizing it into words.

We segment the sentences using **sentence segmentation**. It will be a lot easier to write a program to understand a single sentence than to understand a whole paragraph.

**Tokenization** is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization. Tokenization also helps to substitute sensitive data elements with non-sensitive data elements.

### SENTENCE SEGMENTATION AND WORD TOKENIZATION

✓  
0s

```
[5] nlp = spacy.load("en_core_web_sm")
    doc = nlp(input_text)
    token_words=[]
    #sentence segmentation
    for sent in doc.sents:
        #word tokenization
        for token in sent:
            token_words.append(token.text)
    print(token_words)
```

```
['What', 'is', 'your', 'name', '?', 'I', 'am', 'going', 'home', '.', 'i', 'played', 'with', 'the', 'dog', '.']
```

## Lemmatization

Lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

am, are, is = be

play, playing, played is =play

We have done Lemmatization through 2 methods

- Using normal method
- Using PoS tags

The output of both methods we can see that Pos tagging works better.

## LEMMATIZATION

```
✓ 0s [7] from nltk.stem import WordNetLemmatizer
      from nltk.stem import PorterStemmer
      # Init the Wordnet Lemmatizer
      lemmatizer = WordNetLemmatizer()
```

```
✓ 2s [8] ps = PorterStemmer()
      lemmatized_words=[]
      for w in token_words:
          lemmatized_words.append(lemmatizer.lemmatize(w))
      islsentence = ""
      print(lemmatized_words)
      for w in lemmatized_words:
          if w not in stop_words:
              islsentence+=w
              islsentence+=" "
      print(islsentence)
```

['What', 'is', 'your', 'name', '?', 'I', 'am', 'going', 'home', '.', 'i', 'played', 'with', 'the  
What is your name ? I going home . i played with dog .

```
✓ 0s # Lemmatize with POS Tag
      from nltk.corpus import wordnet

      def get_wordnet_pos(word):
          """Map POS tag to first character lemmatize() accepts"""
          tag = nltk.pos_tag([word])[0][1][0].upper()
          tag_dict = {"J": wordnet.ADJ,
                      "N": wordnet.NOUN,
                      "V": wordnet.VERB,
                      "R": wordnet.ADV}

          return tag_dict.get(tag, wordnet.NOUN)

      # 3. Lemmatize a tokens with the appropriate POS tag
      lem_words=[lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in token_words]
      print(lem_words)
```

```
↳ ['What', 'be', 'your', 'name', '?', 'I', 'be', 'go', 'home', '.', 'i', 'played', 'with', 'the',
```

## Eliminating Stop words and PoS tagging

POS Tagging (Parts of Speech Tagging) is a process to mark up the words in text format for a particular part of a speech based on its definition and context. It is responsible for text reading in a language and assigning some specific token (Parts of Speech) to each word. It is also called grammatical tagging.

### Eliminating Stop words

```
[10] word_set=[]
      for i in lem_words:
          if i not in stop_words:
              word_set.append(i)

      print(word_set)

['what', 'your', 'name', '?', 'I', 'go', 'home', '.', 'i', 'played', 'with', 'dog', '.']
```

### Parts of Speech tagging

```
[11] tagged = nltk.pos_tag(word_set)
      tagged

[('what', 'WP'),
 ('your', 'PRP$'),
 ('name', 'NN'),
 ('?', '.'),
 ('I', 'PRP'),
 ('go', 'VBP'),
 ('home', 'NN'),
 ('.', '.'),
 ('i', 'JJ'),
 ('played', 'VBD'),
 ('with', 'IN'),
 ('dog', 'NN'),
 ('.', '.')]

```



## Sending the processed output to the dataset

### Group of words

INPUT: What is your name? I am going home. I played with the dog.

OUTPUT: ['What', 'your', 'name', '?', 'I', 'go', 'home', '.', 'i', 'played', 'with', 'dog', '.']

We will then use these groups of words to our dataset American Sign Language Lexicon Video Dataset (ASLLVD) that will produce the video output.

<http://dai.cs.rutgers.edu/dai/s/signbank>

Sign video:

What: <http://dai.cs.rutgers.edu/dai/s/video?id=8824&type=separate&datasource=T>

Name: <http://dai.cs.rutgers.edu/dai/s/video?id=5998&type=separate&datasource=T>

### Search for Sign - [HOW TO](#)

Gloss Text: ☐ Exact Match

English Words:

☐ 1-Handed ☐ 2-Handed ☒ Either

DH-Start ND-Start DH-End ND-End



☐ Include Handshapes

# ASLLRP Sign Bank

## Select Sign Type:

All (Exclude Classifier, Gesture, Fingerspelled) ▼

## Current Signs in Sign Bank

A-LEVEL-BELOW

A-LEVEL-BELOW (2)

\*A-LEVEL-BELOW (THINK+A-LEVEL-BELOW)

A-LOT

A-LOT (33)

(1h)A-LOT (1)

A-OK

A-OK (10)

ns-#AA

ns-#AA (2)

ABANDON

ABANDON (10)

#ABC

#ABC (2)

ABDOMEN

ABDOMEN (1)

ABORTION

ABORTION (6)

(1h)ABORTION (7)

(S)ABORTION (4)

ABORTION\_2 (6)

Search for Sign - [HOW TO](#)

Gloss Text: ☐ Exact Match

hello

English Words:

☐ 1-Handed ☐ 2-Handed ☒ Either

DH-Start ND-Start DH-End ND-End

## Sign Search Results

dai.cs.rutgers.edu/dai/s/video?id=4676&type=separate&datasource=T - Google Chrome

Not secure | dai.cs.rutgers.edu/dai/s/video?id=4676&type=separate&datasource=T

[Close](#)

2381 HELLO



2381 HELLO



▶ 0:04 / 0:08

# Related works

- The American Sign Language Lexicon Video Dataset (ASLLVD) consists of videos of >3,300 ASL signs in citation form, each produced by 1-6 native ASL signers, for a total of almost 9,800 tokens. This dataset includes multiple synchronized videos showing the signing from different angles.<http://www.bu.edu/asllrp/av/dai-asllvd.html>
- American Sign language dataset which consist of multiple images to letters used in ASL.
- Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, 2020, Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCAIT – 2020 (Volume 8 – Issue 15)
- Wazalwar, Sampada & Shrawankar, Urmila. (2017). Interpretation of sign language into English using NLP techniques. Journal of Information and Optimization Sciences. 38. 895-910. 10.1080/02522667.2017.1372136.
- Vardan Gupta, Saumya Sinha, English Text to Indian Sign Language Translator  
[https://www.cse.scu.edu/~m1wang/projects/NLP\\_English2IndianSignLanuguage\\_18w.pdf](https://www.cse.scu.edu/~m1wang/projects/NLP_English2IndianSignLanuguage_18w.pdf)



Thank you