

Language Translator

Chaitanyasai Prasanna Kumar Chimakurthy

cchimaku@uncc.edu

https://github.com/saichaitanya1821/Machine_language_translator.git

ttITCS-5156 Fall 2022

October 12, 2022

1. Introduction:

1.1 Problem Statement:

One of the most basic aspects of being human is having the ability to interact with others. Twenty-two of the 121 spoken languages in the country of India alone are recognized as official. However, the most popular translation tool in the world, Google Translate, only covers 9 Indian languages. One can only imagine how much attention the non-official languages would receive when not even the other 13 official languages are being translated. Are we merely to ignore or write off those who speak non-official languages because they don't speak one of the more common languages? How can we tell these minority language speakers vital information, including how to get a simple thing also with locals. I'm sentencing a minority group to live in danger if we are unable to transmit this crucial information.

For the reason, it is necessary to use accurate and reliable language translators for lesser-used languages. Additionally, a means must be found to identify translation mistakes so they may be learnt from and fixed. Another difficulty is determining whether our translation is accurate without having knowledge of both the source and target languages.

1.2 Motivation:

One of the most extensively used languages in the world is English. However, there are currently 7,011 different languages spoken worldwide. Language translation is essential for bridging the gap between various cultures and economies of the globe as our global interdependence and connectivity increase. Despite being one of the most widely spoken languages, only about 40% of English speakers are native speakers. This would indicate that over 60% are non-native English language learners. Undoubtedly, speaking to non-native English speakers in their own language will improve their response. In essence, we are using a translator to link individuals from many nations and ethnic groups based on their common experiences. The skill levels of those who speak English as a second language need to be considered as well. The staggering truth that just 75% of the world's population speaks any English is another. A translator is necessary to get by every day when traveling to various parts of the world and to fully appreciate the experience. We must examine limits as we enter a world that can be digitally transformed with just one click of a mouse and known as the internet. Geography no longer presents a problem while conducting business abroad. The sole access hurdle would be language, yet big enterprises and organizations cannot afford

to miss out on potential customers or revenue. Therefore, in order to grow their business, companies and mega-businesses need invest in a high-quality translation.

But most crucially, translation is necessary for disseminating fresh facts, ideas, and understanding over the globe. Information can only be transmitted to many cultures and areas in this way. influencing the course of history while doing so. People may learn about various works and broaden their understanding of the world in this way. For instance, Ted speeches are available with subtitles in over 100 languages, allowing viewers from all around the world to watch and learn from the top teachers.

1.3 Open questions in the domain

Understanding the conversational context rather than just translating the words is one of the primary issues with NLP. Since few languages (such as some Indo-European languages, Bengali, Chinese, Japanese, and Korean) are gender neutral, it can be challenging for a translator to deal with these languages since translating gender-specific vocabulary might be challenging. Additionally, the model may occasionally become biased if the training data becomes biased (due to an unequal distribution of different types of data).

The allegation that Google Translator is gender biased is the finest illustration of how these two problems interact and the trust of machine translators is still not same. ^[1]

2. Background:

A computer software known as a machine translator may translate voice or text from one language into another. Machine translation technology has existed since the 1950s. Artificial intelligence and natural language processing have become increasingly popular as a result of improvements in computing capacity. There are four types of machine translators

- Statistical Machine Translation (SMT)
- Hybrid Machine Translation (HMT)
- Neural Machine Translation (NMT)
- Rule-Based Machine Translation (RBMT)

We employed neural machine translation for this project (NMT). NMT uses AI to improve its understanding of that language and learn new words (Austin, The Big Guide to Machine Translation). According to IBM Cloud Education, "Natural language processing (NLP) refers to the field of computer science - and more particularly, the field of artificial intelligence or AI - concerned with enabling computers to comprehend written and spoken language in a manner similar to that of humans. ^[2]

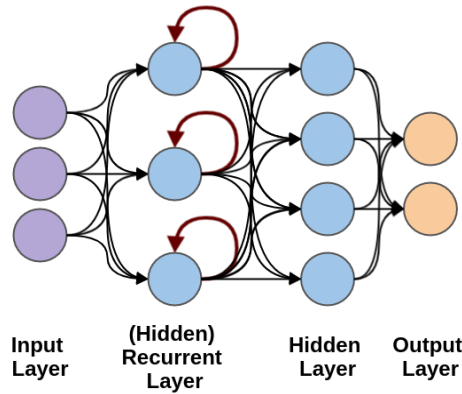
The requirement for enormous volumes of data to train the model is one of the key barriers to natural language processing. Earlier techniques to machine learning were used for natural language processing, and the model was created to respond in a certain way to a set of key words or phrases. However, the method of using deep learning is more beneficial and makes it simpler to grasp the objectives of the speakers by analyzing vast volumes of data, like how a young kid learns to utilize human languages.

Text categorization, text extraction, machine translation, and natural language production are just a few of the activities that may be carried out using natural language processing (NLP). Despite all of these benefits, NLP still faces a number of difficulties, including the collection of vast amounts of data, accuracy in understanding the data, the inability to interpret and comprehend the tone used in language data, as well as the changing nature of language, which results in new definitions and usages of words. [3]

Neural Machine Translation (NMT) has many kinds of machine translations. An artificial neural network is used in neural machine translation (NMT), which contains many different types of machine translations, to predict a chain of numbers when it is given a different chain of numbers. Here, when a word or sentence is provided as an input, it is changed into a chain of numbers in order to be understood, and then by the neural network, it is converted into another chain of numbers that is the needed language that we wanted to be translated into. The fundamental parameters that the neural network uses to translate are created and enhanced by training the neural network using vast linguistic corpora for both the input and output languages. The intricacy of the computations is made possible by the usage of neural networks. Neural networks are extremely flexible and can train highly complicated models on highly complex data because of their many parameters, biases, and weights across nodes. [4]

Seq2Seq generates sentences from input of sentences. It takes a sequence of words in order, translates them into the language you specify, and then outputs a chain of words in the same order. Google first proposed it for automated translation. Seq2Seq performs translation using deep learning, taking into account not just the word being translated at the moment but also the words around it to understand the context of what it is predicting. [5]

We are unable to employ traditional feedforward networks for machine translation since the input data is time series data. Recurrent neural networks (RNNs) are used instead because they have the capacity to retain and utilise historical data in order to forecast future variables. It is possible to train these recurrent neural networks to store historical data. Only data in which the individual data points are independent of one another may be used by normal feedforward neural networks. These RNNs feature the concept of memory, where they retain the information about previous input data so they can produce future output data. The feedback loops in the recurrent neural networks allow them to modify the node weights as necessary to provide the optimal results.



Picture of a Recurrent Neural Network

Recurrent neural networks provide a wide range of benefits and drawbacks, which are detailed below.

Advantages:

- Sequential data can be handled by them.
- They can handle input data of various lengths.
- They can recall information from the past.

Disadvantages:

- The calculations are time-consuming.
- When providing suggestions, the network does not take future input data into account.
- The Vanishing Gradient Problem, in which the weights may approach zero and prevent the addition of additional weights ^[6]

3. Method

3.1 Algorithm and method

To construct a language translator, we cleaned the data, tokenized it, padded it, modeled it, embedded it, encoded it, and decoded it using Long Short Term Memory Networks. We employ Keras as the frontend and TensorFlow as the backend for our framework. As LSTM eliminates vanishing gradients, it is the ideal technique for language translation in our multi-classification model. Compared to other techniques, our algorithm requires a lot of training data, but fortunately our German sample.txt is large enough. By using structured call gates, the algorithm may delete or add information. The sigmoid neural network layer and the pointwise multiplication operator make up these gates. How amount of each component should be allowed through is determined by the sigmoid layer, which generates a value between 0 and 1. Therefore, if the number is 0, nothing passes through, but if it is 1, everything does. The algorithm's outputs are given a probability for each class, which may be utilized with a threshold to lower the number of false positives.

To make the network training as efficient as possible, the following parameters were adjusted:

- Training length (number of epochs)
- Batch size
- Solver type
- Learning rate
- Embedding Size
- Momentum

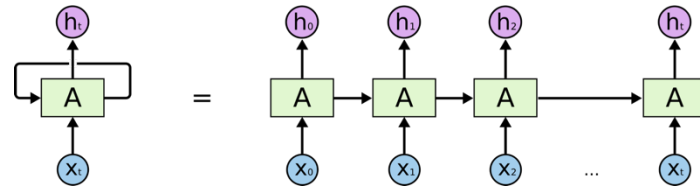
LSTM architecture

- Use Gate
- Forget Gate
- Learn Gate
- Remember Gate

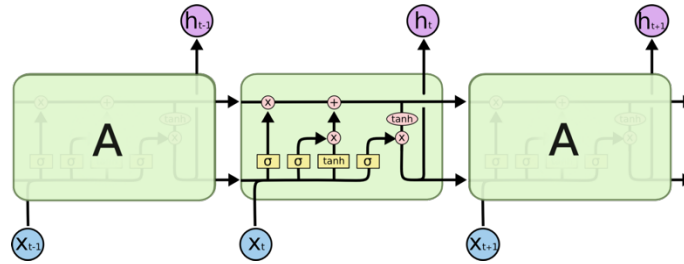
Among the many subfields of computer science and artificial intelligence is natural language processing (NLP) (AI). Giving computers the capacity to comprehend spoken words and text in the same manner that people do is the aim of NLP. NLP integrates statistical, machine learning, and deep learning models with rule-based human language modeling. Computers can process text and human voice using these models and technologies. Recurrent Neural Networks are a popular variation neural network type in NLP (RNN). The normal input for a neural network is the complete sample, but the standard input of an RNN is a word instead. This is how conceptually an RNN varies from a typical neural network.

Due to the fixed structure of a typical neural network, this allows RNN to be flexible enough to operate with phrases of various lengths. Due to this, RNN has an advantage over regular neural networks in that it can share characteristics that have been learnt across various text locations. RNNs are essentially networks with loops that allow information to flow from one phase of the network to the next. Multiple instances of the same network sending a message to the successor is a straightforward way to conceptualize RNN. But one of RNN's drawbacks is that they can't pick up on long-term dependencies.

Long Short Term Memory (LSTM) networks, a unique class of RNNs with the capacity to learn long-term dependencies and prevent vanishing gradients, are here to the rescue. LSTM are designed particularly to retain knowledge for a lengthy period of time, which makes learning easier for them. When inputs and outputs have distinct size grouping, the sequence to sequence model is an unique sort of RNN architecture that is designed to tackle sequence-based issues.



RNN Model



LSTM Model

3.2 Data Preprocessing

The datasets were initially acquired for free from a website (<http://www.manythings.org/anki/>) (www.kaggle.com). For the model, we utilized a dataset from German. Data profiling and data cleaning have first been done on the dataset. The practice of looking through, evaluating, and reviewing data to compile statistics regarding its quality is known as data profiling. While data cleansing is the process of determining the simplest approach to address quality concerns, such as removing inaccurate data, completing data gaps, or generally ensuring the raw data is appropriate for the relevant activity.

The dataset we utilized comprises about 200,000 rows, each with some extra information. To understand the data and what possible cleanup it would need, we personally profiled the data. We have to eliminate some inaccurate or unnecessary data from the dataset as part of data cleansing. We performed data purification since our code specifies that we should only utilize the English and German sentences that match to the sentences in the dataset as inputs.

4. Experiments

4.1 Tokenization:

Tokenization is the act of disassembling a phrase into individual words and then turning those words into numbers or integers that the algorithms can comprehend.

The input list was first tokenized using the built-in function Tokenizer on the token input list. The length of the largest sentence in the input list was then displayed, together with the number of distinct words. The longest sentence in the input list has a length of 6, and there are 3552 unique words total in the input list.

Output Result:

```
➞ Number of the unique words in input list given by me : 3552  
Length of the longest sentence in input list is : 6
```

After tokenizing the input list, we used the provided function Tokenizer to tokenize the combined list of the encoder output list and the input to the decoder list. Then, after merging the encoder output list and the decoder input list, we printed out the number of unique words and the length of the longest sentence from the output list. The longest phrase in the output list was 11 words long, and there were 8090 distinct words total in the output list.

Output Result:

```
➞ Number of unique words in the given list: 8090  
Length of the longest sentence in list: 11
```

According to the results of tokenizing the input and output lists, on the whole, German sentences are longer and include more words than English ones.

4.2 Padding

Padding is the technique of determining a sentence's overall length in advance so that each sentence will be the same length. We must change the sentences into a certain length because the input and output lists' lengths could vary.

By making the longest sentence in the input list equal to all other sentences in the list, we padded the input list. Then we printed the encoder's form for the padded input list and the order for the padded input variable that was previously chosen.

Output Result:

```
➞ Checking shape of padded input sequence for encoder: (20000, 6)  
Padded input sequence for encoder at index [205]: [ 0  0  0  0 63 57]
```

Then we looked up the word indexes in the chosen sentence.

Output Result:

```
63  
57
```

The output list was then padded by using the predefined function pad sequences to the decoder's input sequence. The selected sentence's form and indices from the padded input sequence were then printed.

Output Result:

```
Checking shape of padded input sequence for decoder: (20000, 11)
Padded input sequence for decoder at index [205]: [ 2 398 399  0  0  0  0  0  0  0  0]
```

Then we verified the decoder's indexes for the chosen phrase from the padded input sequence.

Output Result:

```
☞ 2
   398
   399
```

After that we did padding for the output sequences for the decoder. Then we printed the shape of the decoder's padded output sequence as well as the indices of the chosen padded output sequence.

Output Result:

```
☞ examining the decoder's padded output sequence's form: (20000, 11)
Decoder output sequence with padding at index [205] [398 399  1  0  0  0  0  0  0  0  0]
```

4.3 Word Embeddings:

The technique of transforming a word with a vector representation for better analysis is called word embedding. Using Glove (Global Vectors) for word representation, we first constructed the dictionary for the word embeddings before creating the embedding matrix vector. Next, we looked up the embedding value for the previously chosen word in the dictionary.

One Hot Encoding:

Each word in the text data is represented as a vector with the values 1 and 0. A one-hot vector is a vector that is encoded as a single one-hot vector that only contains the values 1 and 0. "This is an Example," for instance. This phrase demonstrates the one hot encoding in the following form.

This: [0 1 0 0 0], Is: [0 0 1 0 0], an: [0 0 0 1 0], example: [0 0 0 0 1]

Because we represent each word and symbol in the text data as a distinct one-hot vector that contains the numerical data 1 and 0, if a word repeated in a phrase, we would assign that word a new encoding. The list of words would be represented as an array of vectors or a matrix as each word is represented as a vector. This allows us to create multi-dimensional vectors that can be fed into neural networks.

Output Result:

```
[ -0.2272    0.41578    0.035985  -0.22895    0.13481   -0.12407
  -0.56653   -0.32096    0.16223   -0.024273  -0.033683  -0.4281
  -0.043623  -0.18682   -0.29002   -0.2464     0.51505   -0.27678
  -0.97691    0.6466    -0.2955   -0.34955    0.0085932  0.17147
  -0.24341    0.018099  -0.57991   -0.68634    0.29723   -0.19539
  -0.39285    0.55418   -0.11855    0.40271   -0.24441    0.13792
    0.14396    0.15116    0.19257   -0.095074  -0.59663   -0.27685
  -0.43814   -0.98947   -0.52515    0.52006   -0.4445   -0.23366
  -0.21019   -0.9059   -0.78404    0.11093    0.09208    1.2145
    0.52067   -2.4821   -0.17693   -0.94136    1.8578    0.17312
  -0.32761    0.45067   -0.092353  0.11201    0.50869    0.52743
  -0.075203  0.73518    0.40716   -0.74727    0.60342   -0.69189
  -0.56087   -0.87252    0.56949    0.35642   -0.25138    0.37888
  -0.68022    0.19444    0.92435    0.5161   -0.62632   -0.16946
  -1.825      0.23406    0.31977    0.76773   -0.83824   -0.38335
  -0.037695  -0.053895  0.1697    -0.97895   -0.5481   -0.25664
  0.10314    0.16628    0.216     0.015172  1.8578    0.17312 ]
```

Next, we examined the embedding value for the previously chosen sample word's word index in the embedding vector matrix.

Output Result:

```
[ -0.2272    0.41578001  0.035985  -0.22894999  0.13481   -0.12407
  -0.56652999 -0.32095999  0.16223   -0.024273  -0.033683  -0.42809999
  -0.043623  -0.18682   -0.29001999 -0.2464     0.51504999 -0.27678001
  -0.97691    0.64660001 -0.29550001 -0.34955001  0.0085932  0.17147
  -0.24341001 0.018099   -0.57990998 -0.68633997  0.29723001 -0.19539
  -0.39285001 0.55418003 -0.11855    0.40270999 -0.24440999 0.13792001
    0.14396    0.15116    0.19257   -0.095074  -0.59662998 -0.27684999
  -0.43814   -0.98947001 -0.52515    0.52006   -0.4445   -0.23366
  -0.21019   -0.9059   -0.78403997 0.11093    0.09208    1.21449995
    0.52067   -2.48210001 -0.17693   -0.94136    1.85780001 0.17312001
  -0.32760999 0.45067   -0.092353  0.11201    0.50869    0.52743
  -0.075203  0.73518002  0.40716001 -0.74726999  0.60342002 -0.69189
  -0.56086999 -0.87252003  0.56949002 0.35642001 -0.25138    0.37887999
  -0.68022001 0.19444001 0.92435002 0.51609999 -0.62632   -0.16946
  -1.82500005 0.23406    0.31977001 0.76773   -0.83824003 -0.38335001
  -0.037695  -0.053895  0.1697    -0.97895002 -0.54809999 -0.25663999
    0.10314   -0.16628    0.21600001 0.015172 ]
```

Then we have created the embedding layer for our input.

4.4 Creating the model:

Choosing our output must be done before we can start building our model. The output array for the decoder output has then been generated. After that, we looked at the form of the final output array.

Output Result:

```
(20000, 11, 8091)
```

The intended output was then produced by iterating over the decoder output sequence using an enumerate function. We chose the enumerate function since it records both the element and the value assigned to the relevant index.

Then, we developed the encoder and the LSTM encoder's internal states. Here, the encoder will receive an English text as input, and it will output the hidden and cell states of the LSTM.

The decoder and the internal states of the LSTM for the decoder were then developed. Here, the decoder receives two inputs from the encoder: the input text and the hidden and cell states. After separating the data, we used the fit function to train the model. 90% of the data will be utilized for training the model, and just 10% will be used for model validation.

Output Result:

```
Epoch 15/20
282/282 [=====] - 115s 407ms/step - loss: 0.6946 - accuracy: 0.8924 - val_loss: 1.4663
Epoch 16/20
282/282 [=====] - 113s 399ms/step - loss: 0.6714 - accuracy: 0.8967 - val_loss: 1.4691
Epoch 17/20
282/282 [=====] - 114s 403ms/step - loss: 0.6536 - accuracy: 0.9001 - val_loss: 1.4872
Epoch 18/20
282/282 [=====] - 113s 401ms/step - loss: 0.6386 - accuracy: 0.9028 - val_loss: 1.4934
Epoch 19/20
282/282 [=====] - 114s 404ms/step - loss: 0.6245 - accuracy: 0.9064 - val_loss: 1.5033
Epoch 20/20
282/282 [=====] - 114s 403ms/step - loss: 0.6108 - accuracy: 0.9093 - val_loss: 1.5249
```

We obtained a training accuracy of around 90.88% and a validation accuracy of approximately 79.41% after 20 epochs. The model appears to be overfitted, as can be seen. Since we are presently utilizing just 20000 samples as our input, we need to add some extra data in order to reduce the overfitting in the model. We can decrease the overfitting of the

model by increasing the amount of samples provided to it. Additionally, we might enhance our model to produce more accurate forecasts. The encoder model remains unchanged.

The outcomes from the model's training and validation are then shown as graphs.

First, using the epochs on the X-axis and the training accuracy on the Y-axis, we created the graph for the model accuracy.

Model 1:

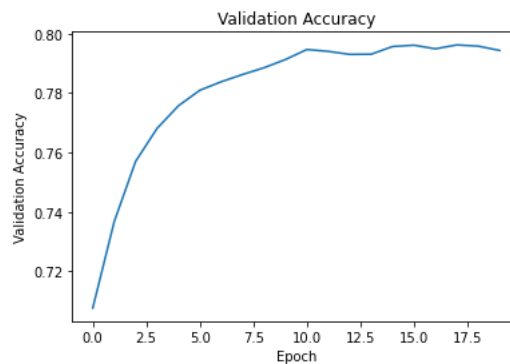
Training:

```
plt.plot(r.history[ 'accuracy' ])  
plt.title( 'Tranining Accuracy' )  
plt.xlabel( 'Epoch' )  
plt.ylabel( 'Accuracy' )  
plt.show( )
```

Accuracy:

```
plt.plot(r.history[ 'loss' ])  
plt.title( 'Training Loss' )  
plt.xlabel( 'Epoch' )  
plt.ylabel( 'Loss' )  
plt.show( )
```

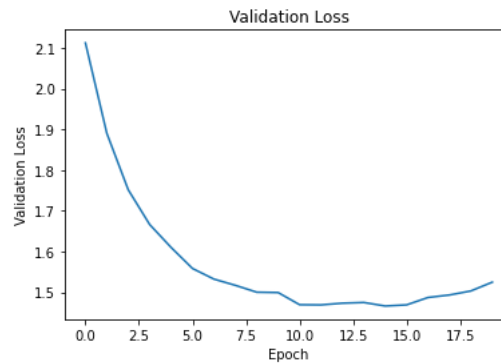
Output Result:



According to the graph for model accuracy, the accuracy increased sharply for the first two epochs before improving almost smoothly after that.

Then, using the epochs as the X-axis and the training loss as the Y-axis, we displayed the graph for the training loss.

Output Result:



From the Training Loss graph, we can see that the percentage of loss decreases fairly steadily after the first two epochs after which there is a sharp decline in the loss.

Then, using the epochs as the X-axis and the validation accuracy as the Y-axis, we generated the graph for the validation accuracy.

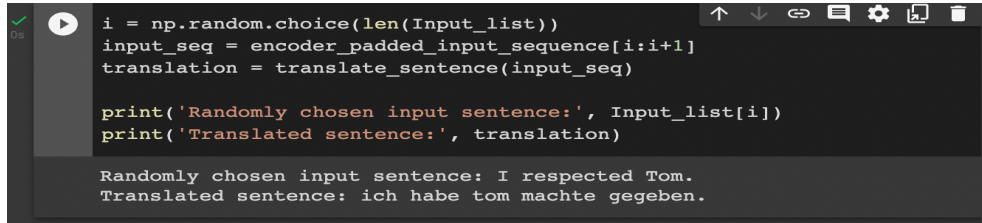
4.5 Translation:

The decoder model must be altered before we can develop a function that can translate the words. The input words will be in English and will be translated into German since we wanted to translate from one language to another. For the final output words—the target words—and the input words, we have built an indexing. Following that, we developed the function `translate sentence`, which is where the term is really translated from English to German. It accepts numerical data as input and outputs the translated German text in the function that was constructed.

4.6 Testing:

We now test the model to see if it is making accurate predictions after modifying the decoder and developing the translation function. In order to achieve that, a random index number must be created, which will choose a random English sentence from the data set and translate it into German. The computer system will choose the sentence at random.

Output Result:



```
i = np.random.choice(len(Input_list))
input_seq = encoder_padded_input_sequence[i:i+1]
translation = translate_sentence(input_seq)

print('Randomly chosen input sentence:', Input_list[i])
print('Translated sentence:', translation)

Randomly chosen input sentence: I respected Tom.
Translated sentence: ich habe tom machte gegeben.
```

With 20,000 observations and 20 epochs in our model (English to German), the validation accuracy was 79.41%.

5. Conclusion

I used recurrent neural networks to create a translation model, and I evaluated the model using several datasets and languages. The results of the translation from English to German reveal that the translation is excellent, and the accuracy is excellent as well.

Since English is the language, we translate into the most frequently, we may see that languages with a script that is similar to English are easier to translate than languages whose scripts are not. Additionally, employing machine translation still faces difficulties in comprehending semantics.

By creating better models and employing richer datasets for model training, the translation efficiency may be increased. By adding a few simple changes to the model, a model may be utilized to translate between multiple languages, improving the translations' accuracy.

To construct the language translator, we employed the best techniques available, including the Seq2Seq methodology and recurrent neural networks. By improving the datasets and adding more characteristics to the model, the translation can still be produced accurately. Additionally, a user interface for the machine translation might have been developed so that it could be integrated into a website and made available to other users.

Limitations:

- Machine translations require a lot of processing power; without it, computations would take a very long time.
- If a code runs at a time, there may be a probability of encountering problems after being updated since the packages used for Natural Language Processing are changed often updated.

Intention to share:

No.

References:

- [1] Ullmann, S & Saunders, D. (2021). "Google Translate is sexist. What it needs is a little gender-sensitivity training." Scroll, 5 Apr, 2021. <https://scroll.in/article/991275/google-translate-is-sexist-and-it-needs-a-little-gender-sensitivity-training#:~:text=Biased%20algorithms,and%20all%20nurses%20are%20female.>
- [2] IBM Cloud Education. "Natural Language Processing (NLP)." IBM, 2 July. 2020, [https://www.ibm.com/cloud/learn/natural-language-processing#:~:text=Natural%20language%20processing%20\(NLP\)%20refers,same%20way%20human%20beings%20can.](https://www.ibm.com/cloud/learn/natural-language-processing#:~:text=Natural%20language%20processing%20(NLP)%20refers,same%20way%20human%20beings%20can.)
- [3] Lutkevich, B. (2021). "natural language processing (NLP)." TechTarget Network. Retrieved May 4, 2022. <https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP>
- [4] Yip, S. (2022). "What is Neural Machine Translation & How does it work?." TranslateFX, 18 Apr, 2022. <https://www.translatefx.com/blog/what-is-neural-machine-translation-engine-how-does-it-work?lang=en>
- [5] Wadhwa, M. (2021). "Seq2seq model in Machine Learning." GeeksforGeeks, 29 Sep, 2021. <https://www.geeksforgeeks.org/seq2seq-model-in-machine-learning/>
- [6] Saeed, M. (2021). "An Introduction To Recurrent Neural Networks And The Math That Powers Them." Machine Learning Mastery, 24 Sep, 2021. <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/>