LANGUAGE TRANSLATOR APPS USING MACHINE LEARNING

Language translator apps, as the name suggests are used to translate texts, from one language to another. They are trained on a large amount of data, using Neural Machine Translation (NMT). So, how do these language translator apps work?
Neural networks basically refers to a machine learning model designed to make the computer perform a certain task, by analyzing training examples.
Neural Machine Translation, uses neural networks to translate the source text (the language which we want to translate) to the target text (the language to which we want to translate). For example, I want to translate a text from English to Hindi, then the text in English will be my source text, and the text in Hindi will be my target text.
The training models used by each one of the translation systems are different, however each one of them perform the same task, that is, language translation.

Let us look at the steps involved

So now, we know the very basic layout of how the language translation happens by using Neural Machine Translation. But what actually are the steps involved in doing so?

1. Preprocessing:

   In the real world, we have a huge data and preprocessing the data is very important in cleaning and filtering the texts we want to translate, and to prepare it for training. Preprocessing includes:

   - Acquiring the dataset: This is the first step of data preprocessing, to build a good model, we need to acquire the relevant dataset, in this case, the dataset of conversion of the source text into the target text.
   - Importing all the libraries: Importing all the necessary libraries required to build the model is very important, as it will help us to use the necessary functions required to build the model.
   - Importing the dataset: Importing the dataset is very important, after which we will be able to access the data values from the dataset for further simplification. (uploading the relevant csv files)

- Identifying the missing values: Many times, our dataset has certain values like "NaN", which indicates, that the particular value in the dataset is missing. We can either remove the rows and columns having such missing values, or we can replace it using some inbuilt functions of the libraries.
- Encoding data: In the case of language translators, our computer would not understand the sentences and words we want to translate, thus there are certain methods by which we can encode the input into vectors/numbers which the computer understands.
- Splitting the dataset: We need to split our dataset based on training and testing. Some percentage of our model, lets say our model uses 80% of data present in the dataset, whose output is known to us. Then the remaining 20% is the test set, which is used for testing the machine learning model.

2. Preparing text which we want to translate for further processing
    - Removing the NaN values: Remove all the missing values from the dataset, and replace them if required.
    - Parsing the dataset file: We will add '\t' to separate input and target texts from the row. And we will add '\n' to separate the rows. The reason is because the computer does not know which word starts a particular sentence and which one ends it. Thus, adding '\t' at the start and adding '\n' to the end will help to tell the model that it is the starting and ending of that text.
    - Removal of trailing white spaces: In the source text and the target text present in the dataset, we should firstly remove all the trailing white spaces.
      For example we have two sentences:
      " I love machine learning. "
      "I love machine learning."
      So, both the above sentences have the same meaning. We, as humans know this, but does the computer know this? No, it doesn't. That is why it is really important to remove the trailing spaces before and after the ending of each sentence, both in the source text and the target text inside the dataset.
    - Removal of punctuation: Punctuation should be removed otherwise the computer wont be able to differentiate between two same words with different punctuations in it.
    - Converting the text to lowercase: It is mandatory to convert the entire text in lowercase. To make it more clear, let me take an example:
      "Hello"
      "hello"
      So here, we have two words with the same meaning, but one starts with capital H and the other one starts with lower h. We as humans know that "hello" and

"Hello" means the exact same word, but our computer will treat it as two separate words. To eliminate this kind of error, we convert the entire text into lowercase.

So, these were the few basic ideas that would help us in preprocessing the dataset. Now, let us look at the techniques which are used to convert the words between languages.

How do these apps maintain the meaning of the words to be translated?

There are a few types of the neural networks which can be used for language translation such as

1. RNN- Recurrent Neural Network
2. LSTM- Long Short Term Memory (type of RNN)
3. GRU- Gated Recurrent Unit (type of RNN)

However, for language translators, LSTM has shown to provide the best results.

We will now see the implementation of each one below:

1. RNN- Recurrent Neural Network
   The recurrent neural network is a type of a neural network, whose output from the previous step is fed as an input to the current step. In a language translator, it is important for the model to have a knowledge of the previous words used in a sentence too. How? Think! The vocabulary is very vast and different usages of the same words placed differently in a sentence could have different meanings. Thus, for translation, to predict the next word of the sentence, the previous words are required. RNN has a feature called the "hidden state", which remembers some information about a previous sequence. Great right! The hidden state is also known as the memory state since it remembers the previous inputs to the networks. So each unit in a RNN has an internal state, which is called the hidden state of the unit. The hidden state signifies the past knowledge that the network holds. This hidden state, is updated at each time to edit the changes of the network about the past.
2. LSTM- Long Term Short Memory
   It is the most used version of RNN. It has tackled the problem of RNN, where it cannot predict the word stored in long term memory, but can give accurate predictions about recent information. LSTM networks are capable of learning long time dependencies of sequential data like in the language translators for predicting the next word based on

the previous input of the word. LSTMs have a memory cell which is capable of long term storage of information.

3. GRU- Gated Recurrent Unit Networks
   GRU is a type of RNN, that can process sequential data such as text, speech and time series data like that of a language translator. It selectively updates the hidden state of the network at each time step. It has two gates- reset and update, which determines how much of the past knowledge to forget, and how much of the past knowledge to retain, respectively.

Implementation of the model (the actual technique behind language translation)

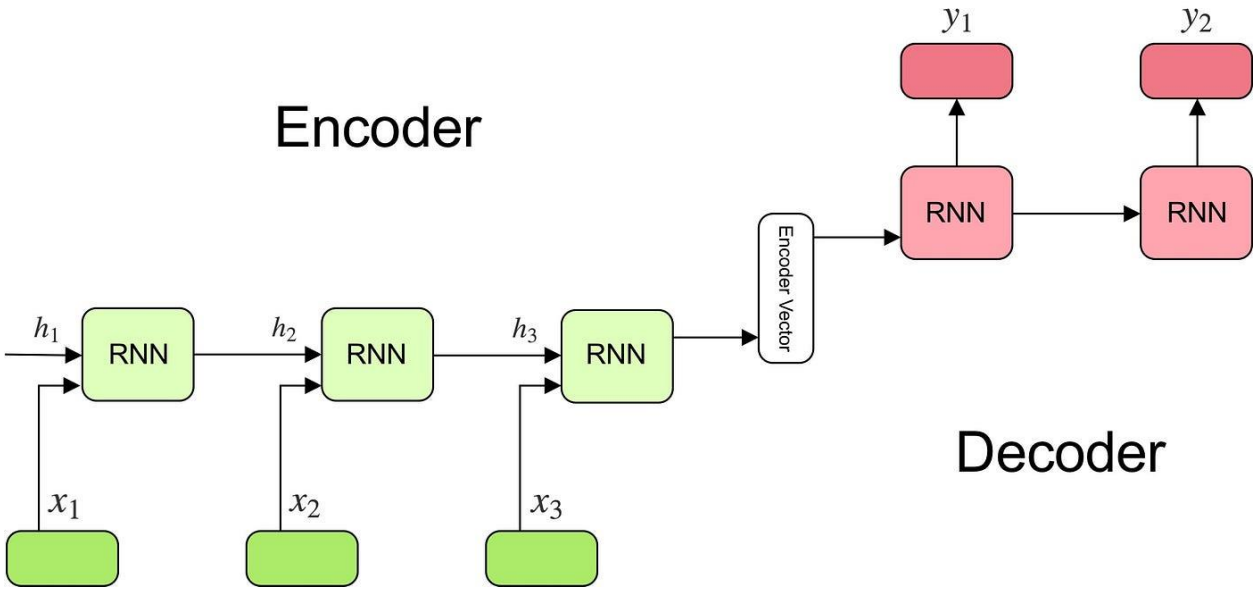
So, we have seen some ways by which the meaning of the sentence would fairly be predicted while translation. However, let us look at the exact model which we will be implementing to carry out the language translation.

So again, there might be many models performing the same task to obtain translation of languages. The model we will be looking at, is the Sequence-to-Sequence model (seq2seq model).

Sequence-to-Sequence model

So, how does sequence to sequence actually work?

In sequence to sequence, there are two main components. One is encoders, and the other one is decoders.



Here, in place of RNN, we can have LSTM or GRU any one. But we usually take LSTMs, as they work well. Here, x1,x2,x3 are words passed at time t, t+1,t+2 and so on. And then we get the output- shown as the encoder vector in the above diagram I am referring to. It is called as a vector because in language translation, I have passed each and every text as a vector. We can use one hot encoding also to represent vectors. (One hot encoding- represents each word present in the sentence as 1, and the words not present as 0, which helps to access the text from the huge dataset). So basically, we convert the text into a vector format and then we pass it through the neural networks. (We don't have any output on the encoder side, that is how it works). Now, once we have the encoder vector, it is then passed to the decoder as a input. We will get an output, that particular output (y1 here) will be the next input for y2. Then we will continue so on and y2 will be an input for yn (considering it occurs n times for n words), giving us an output (which will be the end of the string). Which means it will be the end of the output. The input sequence and the output sequence may be completely different in size.

But now, while doing the prediction for the new test data, we will find the probability of the new text, given the sequence of characters x1,x2,x3 passed. We will take a sequence of inputs and we'll try to find the probability of the new text to be formed (which will contain all the combinations of the new words in the predicted text, based on the inputs). So, this is basically

how a given source text is converted into the target text, using the Sequence-to-Sequence (seq2seq) model.

Improvement of current methods in language translator apps

To think of how to improve the current methods in language translator apps using machine learning, I should first think about the disadvantages of the same.

Let us now look at the drawbacks of language translator apps using machine learning:

1. Satire: According to me, it would be difficult for the language translator to understand humor/satire and the different figures of speech like hyperbole, metaphor.
2. Emojis: The use of emojis in certain texts would not correctly translate the language as per what it means.
3. Extremely difficult vocabulary: Difficult vocabulary like use of higher and complicated words of a certain language would not be able to completely translate that language, leaving it as a drawback.
4. Texts based on contexts: Certain contexts having certain sentences would not mean the same thing if used in a different context. The language translator would not understand such contexts, hence causing a confusion in translation.
5. Using slangs: Certain slangs might not be translated correctly with the usage of abbreviation of words as used by today's generation.
6. Use of idioms: The language translator would not be able to understand the meaning of idioms, and it would rather translate it directly or simply, leaving it as a drawback.

Now, let us look at some ways by which these drawbacks can be improved:

1. We can implement certain databases, which contains words related to humor and build models which can identify such humorous and witty content, and can translate the same.
2. For emojis, they look like images, but they are not. They are basically characters or letters from the Unicode character set. So, we can make sure that the model has Unicode character encoding, which can understand the meaning of emojis used in text. Implementation of a model which can understand both, normal text as well as images, is important so that the meaning of emoji in that particular text can be understood.

3. To simplify difficult vocabulary, we can have an extensive dataset containing words which are a little complex to frame. We can then train our model on this dataset, and eliminate the ambiguity due to extremely difficult vocabulary.
4. Implementation of models, that could translate each paragraph at a time, rather than each sentence, would give more idea about the context. This would ensure that the ambiguity related to context in language translation gets removed.
5. Training our model on certain created datasets containing abbreviations used in texting would make our model understand how to translate abbreviations also, which are commonly used in texting these days, worldwide.
6. Creating a specific dataset related to idioms, and training our model on that, would be a good idea in the case to eliminate ambiguity due to idioms. Our model would understand that this particular phrase should not be translated normally and literally, but it should be translated as we would translate an idiom. Creating a separate dataset for the same would help.