Task 3

Q) How do you think Language Translators Apps use Machine Learning? Explain what preprocessing (cleaning like removal of certain words, text shortening etc.) occurs and how these apps maintain the structure and meaning of the sentence while translating. Describe the techniques used to convert the words between languages. Brownie points if you can think of how, you can improve current methods.

Language Translator Apps extensively use Machine Learning (ML) techniques to provide accurate and contextual translations between languages. This process includes stages like preprocessing, translation and post processing.

The various text preprocessing steps are:

1. **Lower casing**: It is essential for us to convert all the words from our data frame to lowercase as the machine may interpret words such as Case and case as two different words even though they mean the same, it helps to maintain the consistency flow during the NLP tasks. We achieve this by using the lower() function.
2. **Tokenization:** Tokenizing is like splitting a whole sentence into words. We do it so we can analyse each and every word of the data set individually. Using join() function to combine all the text that needs to be processed, together. We can split this combined data by two methods. We can use the inbuilt split() function to separate each individual word and make it a part of a list. Second method is two import the word_tokenize() function to split up the sentences we joined to form the list of tokens(words) we desire.
3. **Stopwords removal:** English is one of the most common languages, especially in the world of social media. For instance, “for”, “the”, “is”, “an” etc. are in the set of most commonly used words. Removing these words helps the model to consider only key features. These words also don't carry much information.

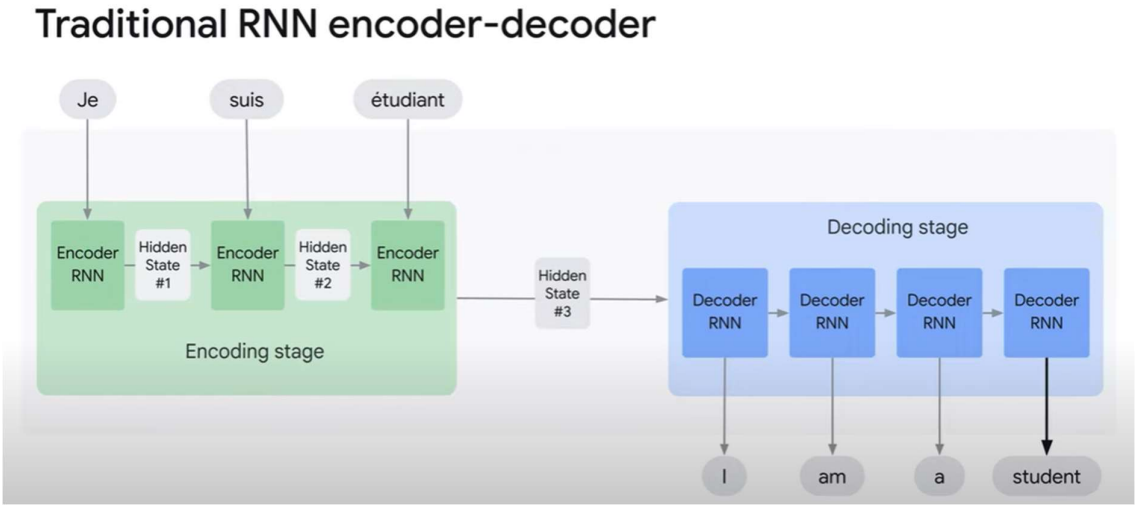
```
words = [word for word in words if word not in set(stopwords.words('english'))]
```

In this statement, we're changing the list "words" which contained the set of all the tokenized words in the data frame and given the condition "if the current word, is not a part of the given Stopwords library, then it can stay in the list, else don't include it" this helps us modify our tokenized word list by getting rid of the stop words.

4. **Stemming:** There are many variations of words that do not bring any new information and create redundancy, ultimately bringing ambiguity when training machine learning models for predictions. Take "He likes to run" and "He likes running" for example. Both have the same meaning, so the stemming function will remove the suffix and convert "walking" to "walk."
5. **Lemmatization:** lemmatization performs normalization using vocabulary and morphological analysis of words. Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word. Lemmatization uses a dictionary, which makes it slower than stemming, however the results make much more sense than what you get from stemming.

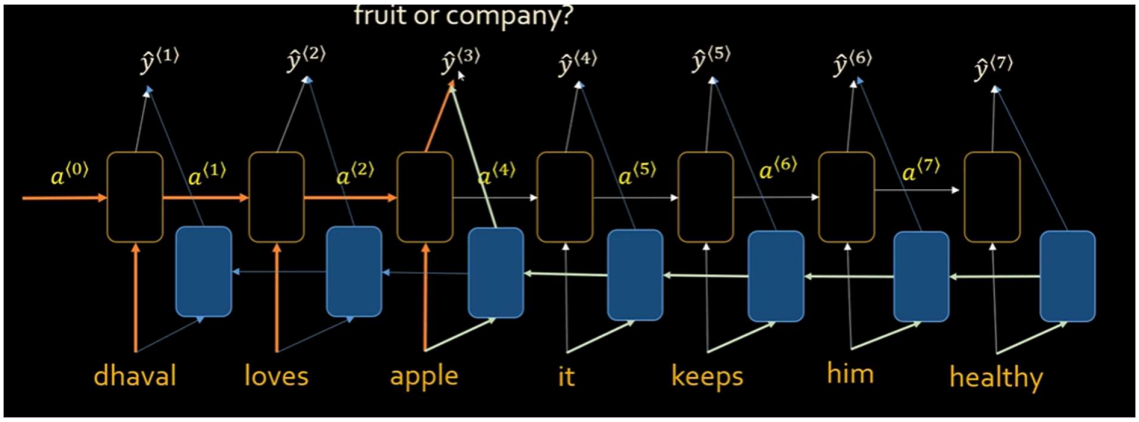
Translation: Now that the text is clean and ready to use. We can move on to the next step which is the translation process. While considering translation we need to keep in mind two major factors that contribute to a language or a sentence. That is the token(words) and the grammar(context). As we know the computer itself does not understand any language except numbers. So, to address that issue we use an embedder that converts the word into a corresponding vector of real numbers. This vector is then passed to a seq2seq encoder decoder model. We use Recurrent neural networks consisting of Long Short term memory networks. The RNN however by itself is slightly inefficient due to the vanishing/exploding gradient problem. Hence, we add something called Long Short term memory networks. Since the major issue with the RNN

is that it uses the same feedback loop with the same weight for all the inputs in particular network, Long Short Term memory uses two separate paths. One for the Long term memory and one for short term memory. As we move through different inputs for the given dataset, the LSTM model keeps updating itself according to the new input values it receives in contrast to RNN which used the same values over and over again. Finally the last value that we receive for the short term memory, is the actual output that we consider for the desired prediction. Then we use an attention mechanism which focuses on specific parts of the input. which helps us to see the corresponding translation for the input and gives a desirable output. This output is then sent to the decoder which is another recurrent neural network that generates the respective translated words, one at a time and provides a sensible output.

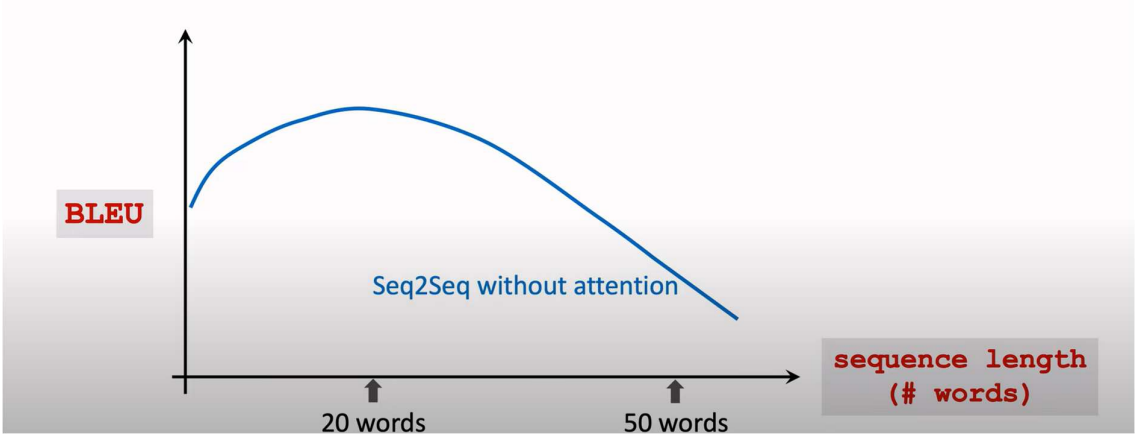


In the above diagram we can see how an encoder-decoder RNN actually works. Another addition to the RNN is that we take it in the form of a bidirectional RNN. As we know, the meaning of the word in a sentence depends not only on the words before it, but also on the words after. Hence, we need to check the output from the words following the particular word too. From the diagram below we can see that the word apple in this statement can be taken in two ways. We can be talking about apple the fruit or apple the company, which is no longer considered as a proper noun due to the lower case pre processing

step. So in order to give the proper output for y3, we need both, the contextual value from the returned output from y7 and the initial output from y3 to check which word it is.



However the problem with these is that they only work on short sentences and not long ones. Hence increasing inaccuracies in translation whenever the sentence size increases above a certain level. Even with the long short term memory, for longer sentences the initial words that were sent as inputs maybe forgotten as only the final hidden state is being passed as an input for the decoder. As shown in the figure below. The BLEU score (indicating how good the translation is) decreases after a length of 20 words.



We fix this issue by adding an attention mechanism to our model. The attention mechanism in this case is a unit that helps us by making sure that the RNN seq2seq doesn't forget any source input and helps the model better understand what parts to focus on. It helps us better analyse the context of the sentence

given as the input to provide a more accurate output. Attention mechanism checks the usage of the token in 1000s of different sentences, and checks the words around it, trying to see which is more suited for the current sentence given as an input. Accordingly, it gives out a suitable output using the words which it thinks better suits the purpose of the input, and sends the appropriate matrix out to the decoder so it gives us a proper word in the desired language.

After the translation we can apply post processing to make the translated text clearer by applying punctuations, changing the case of the letters etc.

Ways to make Translation better:

1.) **Contextual Understanding:** Incorporating even more sophisticated contextual understanding can help the model capture nuanced meanings and improve translations, especially for idiomatic expressions and culturally specific phrases.

2.) **Fine-tuning:** Fine-tuning models on specific domains or industries can lead to more accurate translations in those domains, as the model becomes more attuned to the relevant terminology and context.

3.) **User Feedback Integration:** Collecting feedback from users and integrating it into the training process can help the model learn from real-world usage and improve over time.

4.) **Multi-modal Integration:** Incorporating visual context from images or video alongside textual input can further enhance the accuracy and relevance of translations.