Language Translator

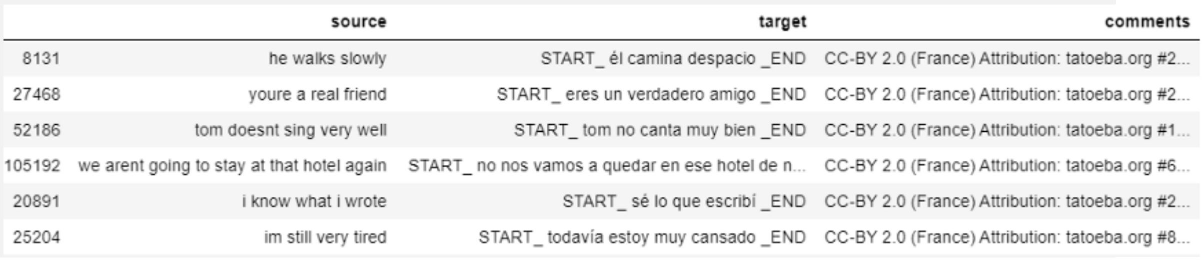
Introduction

In this project we are building a neural machine translator using English as the source language and Spanish as the target language. We will be using Long Short Term Memory(LSTM) units in keras.

You will learn how to:

- Tag the start and end of the target sentence using START_ and _END respectively for training and inference
- Create the dictionary of unique source and target words to vector and vice-versa
- Shuffle the data for better generalization
- Split the dataset into train and test data
- Create the data; we will be using fit_generator() to fit the data to the model
- Build the encoder using Embedding and LSTM layers
- Build the decoder using Embedding and LSTM layers and takes input from the embedding layer and the encoder states.
- Compile the model and train the model.

Add the START_ and the _END tags to the target sentences.

- Adding the START_ and the _END token to the target sentences is very useful for training and during inference. These tags help to know when to start the translation and when to end the translation.
-
```
# Add start and end tokens to target sequences
lines.target = lines.target.apply(lambda x : 'START_ '+ x +
' _END')
lines.sample(6)
```
- 

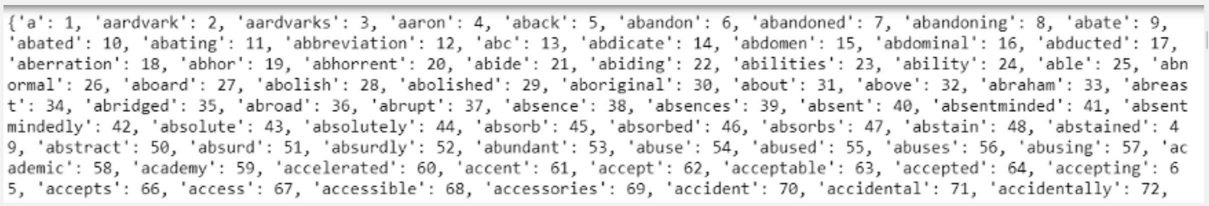
	source	target	comments
8131	he walks slowly	START_ él camina despacio _END	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
27468	youre a real friend	START_ eres un verdadero amigo _END	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
52186	tom doesnt sing very well	START_ tom no canta muy bien _END	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
105192	we arent going to stay at that hotel again	START_ no nos vamos a quedar en ese hotel de n...	CC-BY 2.0 (France) Attribution: tatoeba.org #6...
20891	i know what i wrote	START_ sé lo que escribí _END	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
25204	im still very tired	START_ todavía estoy muy cansado _END	CC-BY 2.0 (France) Attribution: tatoeba.org #8...

-

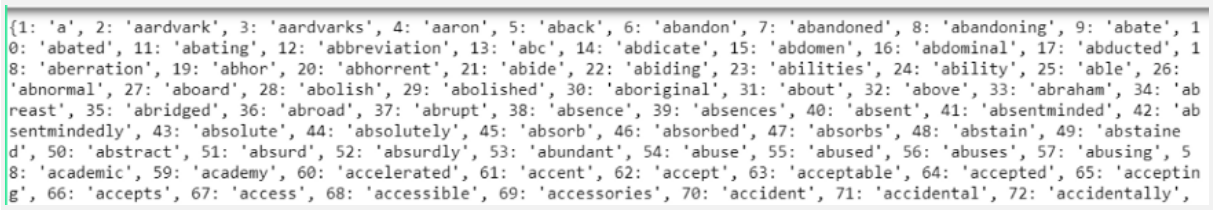
Create a word to index dictionary and an index to word dictionary for all unique source and target words in the dataset.

Size of the word to vector will be based on the length of the source and target vocabulary

```
# creating a word to index(word2idx) for source and target
source_word2idx= dict([(word, i+1) for i,word in
enumerate(source_words)])
target_word2idx=dict([(word, i+1) for i, word in
enumerate(target_words)])
```



```
{'a': 1, 'aardvark': 2, 'aardvarks': 3, 'aaron': 4, 'aback': 5, 'abandon': 6, 'abandoned': 7, 'abandoning': 8, 'abate': 9,
'abated': 10, 'abating': 11, 'abbreviation': 12, 'abc': 13, 'abdicate': 14, 'abdomen': 15, 'abdominal': 16, 'abducted': 17,
'aberration': 18, 'abhor': 19, 'abhorrent': 20, 'abide': 21, 'abiding': 22, 'abilities': 23, 'ability': 24, 'able': 25, 'abn
ormal': 26, 'aboard': 27, 'abolish': 28, 'abolished': 29, 'aboriginal': 30, 'about': 31, 'above': 32, 'abraham': 33, 'abreas
t': 34, 'abridged': 35, 'abroad': 36, 'abrupt': 37, 'absence': 38, 'absences': 39, 'absent': 40, 'absentminded': 41, 'absent
mindedly': 42, 'absolute': 43, 'absolutely': 44, 'absorb': 45, 'absorbed': 46, 'absorbs': 47, 'abstain': 48, 'abstained': 4
9, 'abstract': 50, 'absurd': 51, 'absurdly': 52, 'abundant': 53, 'abuse': 54, 'abused': 55, 'abuses': 56, 'abusing': 57, 'ac
ademic': 58, 'academy': 59, 'accelerated': 60, 'accent': 61, 'accept': 62, 'acceptable': 63, 'accepted': 64, 'accepting': 6
5, 'accepts': 66, 'access': 67, 'accessible': 68, 'accessories': 69, 'accident': 70, 'accidental': 71, 'accidentally': 72,
source_idx2word= dict([(i, word) for word, i in 
source_word2idx.items()])
print(source_idx2word)target_idx2word =dict([(i, word) for word, 
i in target_word2idx.items()])
```



```
{1: 'a', 2: 'aardvark', 3: 'aardvarks', 4: 'aaron', 5: 'aback', 6: 'abandon', 7: 'abandoned', 8: 'abandoning', 9: 'abate', 1
0: 'abated', 11: 'abating', 12: 'abbreviation', 13: 'abc', 14: 'abdicate', 15: 'abdomen', 16: 'abdominal', 17: 'abducted', 1
8: 'aberration', 19: 'abhor', 20: 'abhorrent', 21: 'abide', 22: 'abiding', 23: 'abilities', 24: 'ability', 25: 'able', 26:
'abnormal', 27: 'aboard', 28: 'abolish', 29: 'abolished', 30: 'aboriginal', 31: 'about', 32: 'above', 33: 'abraham', 34: 'ab
reast', 35: 'abridged', 36: 'abroad', 37: 'abrupt', 38: 'absence', 39: 'absences', 40: 'absent', 41: 'absentminded', 42: 'ab
sentmindedly', 43: 'absolute', 44: 'absolutely', 45: 'absorb', 46: 'absorbed', 47: 'absorbs', 48: 'abstain', 49: 'abstaine
d', 50: 'abstract', 51: 'absurd', 52: 'absurdly', 53: 'abundant', 54: 'abuse', 55: 'abused', 56: 'abuses', 57: 'abusing', 5
8: 'academic', 59: 'academy', 60: 'accelerated', 61: 'accent', 62: 'accept', 63: 'acceptable', 64: 'accepted', 65: 'acceptin
g', 66: 'accepts', 67: 'access', 68: 'accessible', 69: 'accessories', 70: 'accident', 71: 'accidental', 72: 'accidentally',
```

Shuffle the data

Shuffling helps with

- Reducing variance
- Ensures models remain generic and overfit less
- Batches between epochs do not look alike
- Makes model more robust

```
#Shuffle the data
lines = shuffle(lines)
```

Creating training and test dataset

```
# Train - Test Split
X, y = lines.source, lines.target
X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size = 0.1)
X_train.shape, X_test.shape
```

Create data for training the encoder-decoder model.

We will use fit generator() instead of the fit() method as our data is too large to fit into the memory. Fit generator is used to generate the data.

```
num_encoder_tokens=len(source_words)
num_decoder_tokens=len(target_words) +1
```

Define the model

The sequence to sequence model takes encoder and decoder inputs to output decoder outputs

```
# Define the model that takes encoder and decoder input
# to output decoder_outputs
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

Train the model

To train the model, we first compile the model and then fit the data to the model

We compile the model using “rmsprop” optimizer; use categorical_crossentropy as we use categorical labels which is one-hot encoded vectors

```
model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', metrics=['acc'])
```

setup the parameters to fit the model

```
train_samples = len(X_train) # Total Training samples
val_samples = len(X_test)    # Total validation or test samples
batch_size = 128
epochs = 100
```

CONCLUSION:

Language translation is most important to feature a person can ask so that when we are in different place we can encouter the language problem.

For communication we should know all the languages which is impossible by writing the simple code we can get rid of this problem.

REFERENCE:

The dataset is taken from the internet that is English to Spanish translation dataset.