internship studio

# Internship Project Solution Document

## Language Translation (German to English)

The computers' emerging ability to understand and analyse human language is Natural Language Processing. In general, the steps involved are:

1. Pre-processing: Tokenization and UNK Replacement
2. Word Embeddings: Vectors and Dimension Reduction
3. Sequence to Sequence: Encoding and Decoding

**Steps break down for project on Language Translation:**

1. Import the Required Libraries

2. Read the Data into our IDE

Our data is a text file (.txt) of English-German sentence pairs.

First, read the file using functions and then need to define another function to split the text into English-German pairs separated by '\n'.

They then split these pairs into English sentences and German sentences respectively. The actual data contains over 150,000 sentence-pairs.

The data we work with is more often than not unstructured so there are certain things we need to take care of before jumping to the model building part.

- Text Cleaning
  We will get rid of the punctuation marks and then convert all the text to lowercase.
- Text to Sequence Conversion
  A Seq2Seq model requires that we convert both the input and the output sentences into integer sequences of fixed length. But before we do that, let's visualise the length of the sentences. We will capture the lengths of all the sentences in two separate lists for English and German, respectively - the maximum length of the German sentences is 11 and that of the English phrases is 8.
- Next, vectorize our text data by using Keras's *Tokenizer ()* class. It will turn our sentences into sequences of integers. We can then pad those sequences with zeros to make all the sequences of the same length.
- Note that we will prepare tokenizers for both the German and English sentences:

**MODEL BUILDING:**

1. Split the data into train and test set for model training and evaluation, respectively.

2. Start off by defining our Seq2Seq model architecture:

- For the encoder, we will use an embedding layer and an LSTM layer
- For the decoder, we will use another LSTM layer followed by a dense layer

3. Use the RMSprop optimizer in this model as it's usually a good choice when working with recurrent neural networks.

4. Use 'sparse_categorical_crossentropy 'as the loss function. This is because the function allows us to use the target

sequence as is, instead of the one-hot encoded format. One-hot encoding the target sequences using such a huge vocabulary might consume our system's entire memory.

5. Train model for 30 epochs and with a batch size of 512 with a validation split of 20%.

   80% of the data will be used for training the model and the rest for evaluating it. You may change and play around with these hyperparameters.

6. Also use the Model Checkpoint () function to save the model with the lowest validation loss.

7. Predictions that will be obtained will be sequences of integers. We need to convert these integers to their corresponding words.

8. Convert predictions into text (English).

9. Finally put the original English sentences in the test dataset and the predicted sentences in a data frame.

10. We can randomly print some actual vs predicted instances to see how our model performs.