Russian-English Language Translation using Deep Learning

Introduction:

The ability to communicate with on another is a fundamental part of being human. Our world is increasingly connected with nearly 7,000 different languages, language translation provides a critical cultural and economic bridge between people from different countries and ethnic groups. Today, expert systems powered by Artificial Intelligence are high in demand that can be used for such linguistic purposes. Indeed the pay-scale of foreign business translators and stenographers is skyrocketing. So, this is a subtle hint which demands our developers for a prototype that is challenging, smart, and beneficial. Some of the use-cases include: Business, Commerce, Media, Education, and Governance. To meet these needs tech giants are heavily investing in machine translation. According to Google, switching to deep learning produced a 60% increase in translation accuracy

Project Goal:

In this project, we have built a Deep Neural Network that functions as Language Translation Tool which in turn can mimic “Google Translator”. The model accepts Russian text as input and returns the equivalent English translation. The goal is to achieve the highest translation accuracy possible.

Approach:

To translate the corpus of Russian text to English, we should build a Recurrent Neural Network (RNN). First we download the dataset and create Data Pairs of Russian-English sentences. Then we apply text cleaning and pre-processing techniques to make the data convenient for model building. Later we design the neural network architecture that can solve our specific problem and train the model on train data. For preventing the model from overfitting and under fitting we will add checkpoints. After the model is successfully trained and achieved the accuracy levels we are trying for, we will make predictions on test data and compare the predictions and test data. Finally, we will build a custom prediction function which takes a Russian sentence and give back its English translated sentence.

Recurrent Neural Networks:

RNNs are designed to take sequences of text as inputs or return sequences of text as outputs, or both. They’re called recurrent because the network’s hidden layers have a loop in which the output and cell state from each time step become inputs at the next time step. This recurrence serves as a form of memory and allows contextual information to flow through the network so that relevant outputs from previous time steps can be applied to network

operations at the current time step. Depending on the use-case, you'll want to set up your RNN to handle inputs and outputs differently. For this project, we'll use a many-to-many process where the input is a sequence of Russian words and the output is a sequence of English words.

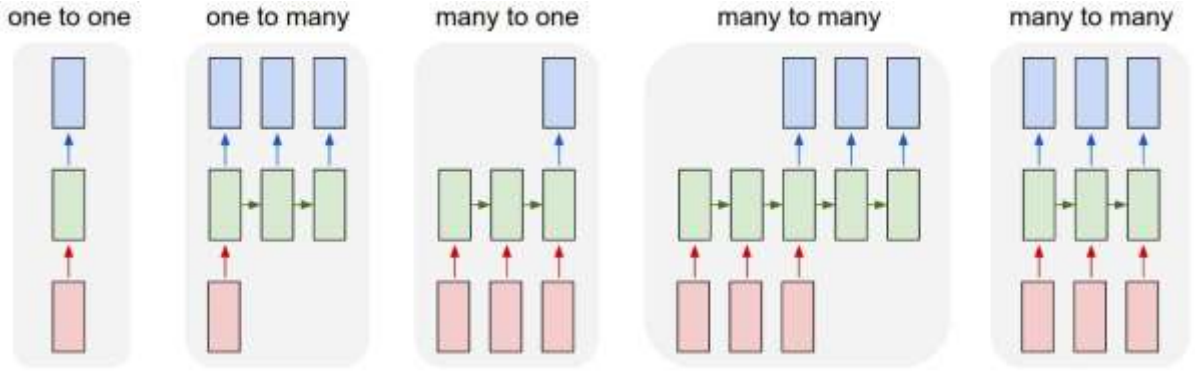

Figure 1: Different Recurrent Neural Networks

Steps Involved in the Project:

The project involves various steps, beginning from reading the dataset file to the end making predictions on custom input. Here is a summary of the various pre-processing and modelling steps. The high-level steps include:

- **Pre-processing:** Load the Dataset, Examine the Data, Separating the Language Data Pairs, and Text transformations like removing punctuations, converting to lower case, tokenization, and padding.
- **Modelling:** Building, Compiling, Training, and Testing the Model.
- **Prediction:** Generate specific translations of Russian to English, and compare the output translations to the ground truth translations, and predicting on custom input sentences.
- **Hyper parameters Tuning:** Iterate on the model, experimenting with different architectures, changing the trainable parameters and looking for better accuracy results.

Model Used:

We are considering Seq2Seq model architecture. We will be using an Embedding layer and an LSTM layer as our Encoder and another LSTM layer followed by a Dense layer as the Decoder. Let's check the architecture of an RNN at a high level. Following is the brief of the layers that are the part of the architecture.

1. **Input Layer:** Input sequences are fed into the model with one word for every time step. Each word is encoded as a unique integer that maps to the Russian dataset vocabulary.
2. **Embedding Layer:** Embeddings are used to convert each word to a vector. The size of the vector depends on the complexity of the vocabulary.

3. **Recurrent Layers (Encoder):** In this layers the context from word vectors in previous time steps is applied to the current word vector.
4. **Dense Layers (Decoder):** These are typical fully connected layers used to decode the encoded input into the correct translation sequence.
5. **Output Layer:** The outputs are returned as a sequence of integers vectors which can then be mapped to the English dataset vocabulary.

Here is the brief summary of the model used:

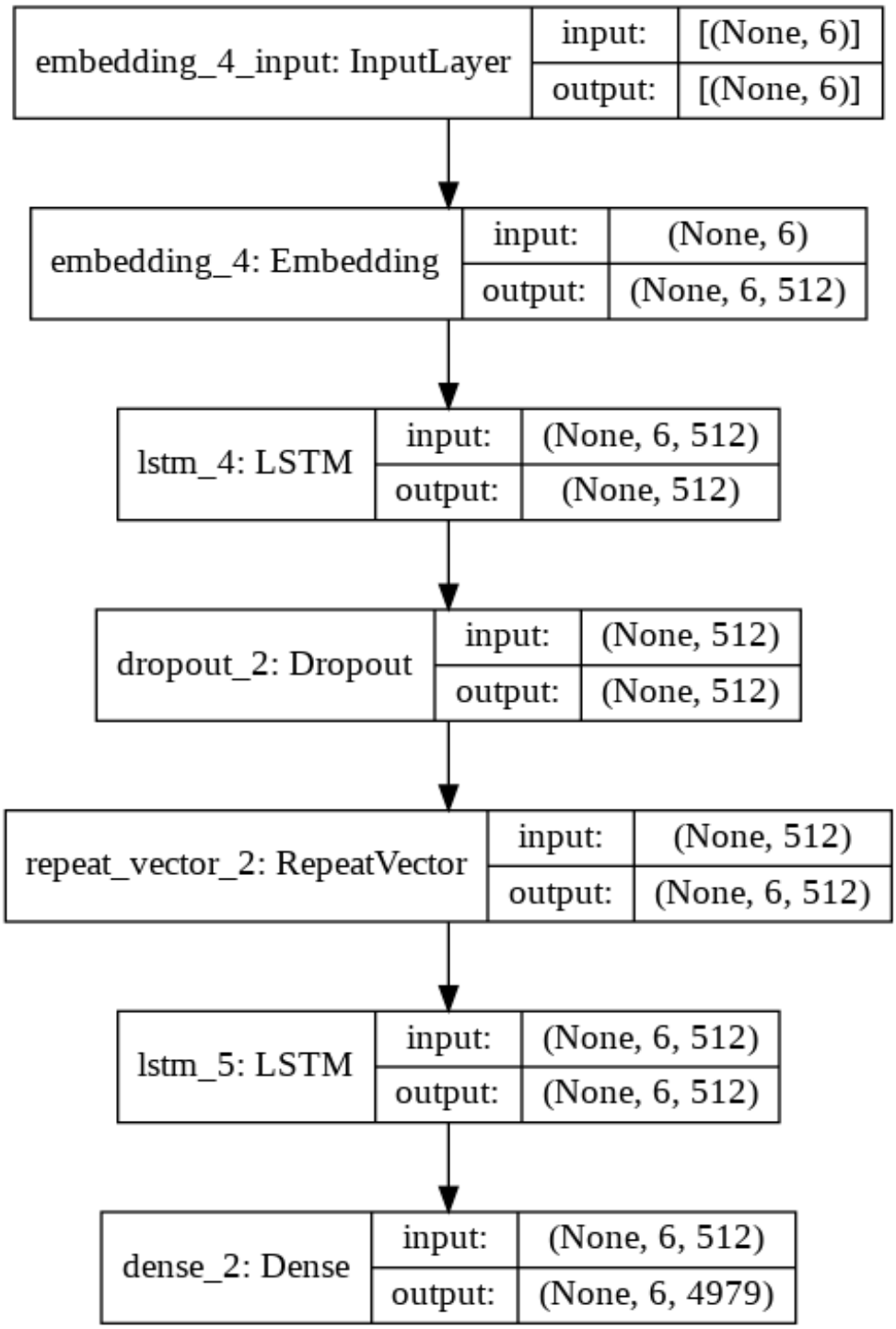

Figure 2: Architecture of the Model

The model was compiled by using the 'RMSprop' optimizer with learning rate of 0.001 and 'sparse_categorical_crossentropy' as loss function. For further reducing the overfitting problem a Dense layer and L1_L2 Regularization are added.

Results:

Here are the Loss and Accuracy metrics for the corresponding training:

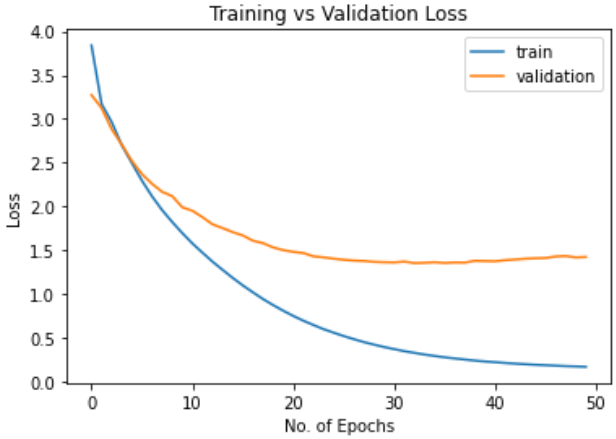

Figure 3: Training vs Validation Loss

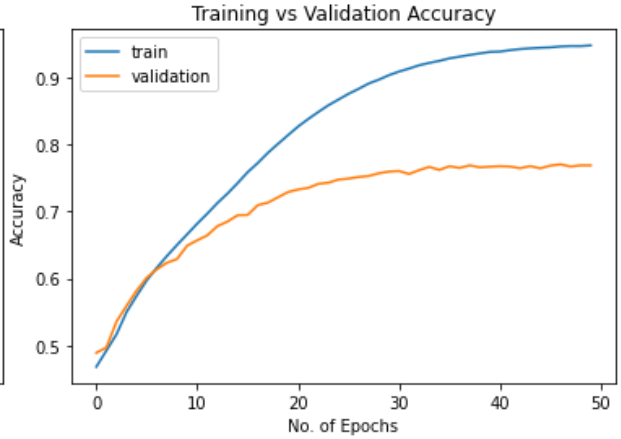

Figure 4: Training vs Validation Accuracy

Future Work:

Initially the model has performed well for 30 epochs with accuracy of 97.25% and validation accuracy of 91.08%. Later upon increasing the batch size and changing the hyoerparameters the model tends to show overfitting characteristics. It is advised to play with the hyperparameters to make the model give best results.