Machine Translator

In this project we will aim to transform english sentences to french sentences using the concept of Machine Translation. The given dataset contains the english sentences and their french counterparts. The data will be cleaned,processed and then tokenized, such that it is ready for machine consumption, by which I mean that it is ready to be feeded into the neural network. The neural network will output the converted sentences.

Built With

- Python
- PyTorch

Dataset

ManyThings.org or directly from Tatoeba into `data/` directory for training and testing.

What We Have Done

We'll look at how to build a language translation model, which is another well-known use for neural machine translation. Using Python's Keras library, we will construct our language translation model using the seq2seq architecture.

1) Data Preprocessing The seq2seq architecture is frequently used as the basis for neural machine translation models. The encoder LSTM and the decoder LSTM networks make up the encoder-decoder architecture known as the seq2seq. The sentence in the original language serves as the input for the encoder LSTM, and the sentence in the translated language along with a start-of-sentence token serves as the input for the decoder LSTM. With a token at the end of the sentence, the output is the actual target sentence.

2) Tokenization and Padding After tokenizing the original and translated sentences, padding is applied to any sentences that are either too long or too short. In the case of inputs, this padding will be equal to the length of the longest input sentence. Additionally, the longest sentence in the output will be this length. 3) Word Embeddings We must transform our words into their corresponding numeric vector representations because we are using deep learning models, and deep learning models only work with numbers. However, we have already transformed our words into integers. What distinguishes word embeddings from integer representation, then? Word embeddings and single integer representations differ primarily in two ways. A word is only represented by a single integer in integer representation. A word is represented as a vector in vector representation, which can have any number of dimensions—50, 100, 200, etc. Word embeddings therefore record a great deal more information about words. Second, the links between various words are not represented by the single-integer representation. Word embeddings, on the other hand, preserve the connections between the words. You have two options: pretrained word embeddings or custom word embeddings. 4) Creating the Model a) The first thing we need to do is to define our outputs, as we know that the output will be a sequence of words. for each input sentence, we need a corresponding output sentence. b) The final layer of the model, which will be a dense layer for making predictions, requires outputs in the form of one-hot encoded vectors because the dense layer will use the softmax activation function. The next step is to

assign 1 to the column number that corresponds to the word's integer representation in order to produce such one-hot encoded output. c) Next, we need to create the encoder and decoders. The input to the encoder will be the sentence in English and the output will be the hidden state and cell state of the LSTM. d) The next step is to define the decoder. The decoder will have two inputs: the hidden state and cell state from the encoder and the input sentence, which actually will be the output sentence with an token appended at the beginning. e) The model is trained on 18,000 records and tested on the remaining 2,000 records. The model is trained for 20 epochs, you can modify the number of epochs to see if you can get better results. After 20 epochs, I got training accuracy of 89.83% and the validation accuracy of 78.12% 5) Modifying the Model for Predictions. a) Since now at each step we need the decoder hidden and cell states, we will modify our model to accept the hidden and cell. b) Now at each time step, there will be only single word in the decoder input, we need to modify the decoder embedding layer. c) The final step is to define the updated decoder model. 6) Making Predictions. a) Words were transformed into integers throughout the tokenization processes. The decoder will also produce integer outputs. On the other hand, we need a string of French words as our output. We must do this by changing the integers back to words. For both inputs and outputs, we will create new dictionaries with words as the corresponding values and integers as the keys. b) Next we will create a method, i.e. translate_sentence(). The method will accept an input-padded sequence English sentence (in the integer form) and will return the translated French sentence. Look at the translate_sentence() method: 7) Testing the Model. To test the code, we will randomly choose a sentence from the input_sentences list, retrieve the corresponding padded sequence for the sentence, and will pass it to the translate_sentence() method. The method will return the translated sentence.

Contributors


```
Sudhir Sharma 12041500
```

```
Amar Budhiraja   12040100
```

```
Anshu Kumar   12040260
```

```
Hrishik Rajesh Kanade  12040670
```