Capstone Project – NLP Machine Translation

Group 09 - Interim Report

Towards partial fulfillment of PGP-Certificate course in AIML.
Submitted to University of Texas at Austin, McCombs School,
Through Great Learning

Kunal Joshi, Nasmida M., Yokesh M. A. , Gobinath Velusamy, Chaitanya Mehta

Contents

1. Abstract

Recent advances in text processing technologies, and computer algorithms have provided us, fast and linguistically accurate translators. The meteoric rise in computing power, availability of training datasets, and sharper algorithms have made it possible to suggest words and phrases that suit the context and situation.

Here, we focus on the language translation between European language pairs, specifically, neural translation between English and German languages. Research work in Machine Translation (MT) started as early as 1950's, primarily in the United States. These early systems relied on huge bilingual dictionaries, hand-coded rules, and universal principles underlying natural language.

As part of milestone - 1, of the Capstone project, we explored the given data sets from ACL2014 Ninth workshop on Statistical Machine Translation. We have imported and merged the dataset given as three distinct text files, after which we performed exploratory data analysis on it. Some Text Pre-processing steps such as tokenizing texts, creating vocabulary and converting the text into numerical format were performed. We have developed a base Recurrent Neural Network (RNN) model and LSTM model, for the translation of texts given as three distinct text files. The datasets were merged and performed exploratory data analysis on it. In view of the computational resources requirements, we have taken a sample training set of 5% of original records, out of the total 4476191 sentences.

2. Introduction

Most of us were introduced to machine translation when Google came up with the service. But the concept has been around since the middle of last century. In 1954, IBM held a first ever public demonstration of a machine translation. The system had a pretty small vocabulary of only 250 words and it could translate only 49 hand-picked Russian sentences to English. The number seems miniscule now but the system is widely regarded as an important milestone in the progress of machine translation. After a long dry period, in 1981, a new system called the METEO System was deployed in Canada for translation of weather forecasts issued in French into English. It was quite a successful project which stayed in operation until 2001. And then came the breakthrough we are all familiar with now – Google Translate. It has since changed the way we work (and even learn) with different languages.

3. Understanding the Problem Statement

Our objective was to design a Machine Translation model that can be used to translate sentences from German language to English language or vice-versa using a Neural machine translation System (NMT). Here both the input and output are sentences i. e. a sequence of words going in and out of the model prompting us to think of a sequential model. Sequential to Sequential Model has two major RNN models one of which acts as a decoder and other as an encoder.

One of the challenges we faced while reading and merging the three datasets was that the "newscommentary" dataset was imbalanced such that English translations of few German sentences were unavailable. Another one was the datasets though were majorly in their corresponding language , some were corrupted with the presence of duplicates and other languages.

4. EDA and Preprocessing

Data Preprocessing is a technique that is used to convert the raw data into a clean data set.Converted clean data will be the first step to give good accuracy in the model . Also we did basic exploratory data analytics to understand data nature.

4.1 Data Statistics

Parameter	Stats
Europarl English dataset length	1920209
Europarl German dataset length	1920209
Commoncrawl English dataset length	2399123
Commoncrawl German dataset length	2399123
Newscommentary English Dataset length	201995
Newscommentary German dataset length	201854
Null values present in the combined dataset	141
Duplicate values present in the combined dataset	45016
English vocabulary of sampled dataset	39171
German vocabulary of sampled dataset	63718
Maximum Length of English Token from sampled dataset	472
Maximum Length of German Token from sampled dataset	464

4.2 Data Cleaning

- **Null Check**

One way of handling missing values is the deletion of the rows or columns having null values.

News Commentary dataset some rows in English have null values . So we dropped those rows.

Pandas function used : dropna()

- **Duplicate**

Remove duplicate records in the data. This is important because duplicate data can introduce bias and affect the accuracy of machine learning models.

Pandas Function used : drop_duplicates()

4.3 Text Cleaning

Once data cleansed using above steps we created a python function to perform text cleansing to make data more meaningful to the model.

As part of pre-processing, we performed:

- Lowercasing of data
- Removing punctuations
- Removing digits
- Removing extra space
- Removing repetition of words

As this project is for machine translation, we have not removed the stop words from English language because it can impact the translation of English to German or vice versa.

```
def process_data(df):

   transformed_df = df

   transformed_df['English'] =
   transformed_df['English'].str.lower()
   transformed_df['German'] =
   transformed_df['German'].str.lower()

   transformed_df = transformed_df.astype(str).applymap(lambda x:
   str(x.replace('\n','')))

   transformed_df['German'] =
   transformed_df['German'].str.strip()
   transformed_df['English'] =
   transformed_df['English'].str.strip()

   string_punctuation = string.punctuation + "¡" + '¿'
   transformed_df['English'] =
   transformed_df['English'].map(lambda x:
   x.translate(str.maketrans('', '', string.punctuation)))
   transformed_df['German'] = transformed_df['German'].map(lambda
   x: x.translate(str.maketrans('', '', string.punctuation)))

   transformed_df = transformed_df.replace('', np.nan)
   transformed_df = transformed_df.dropna()

   transformed_df['English'] =
   transformed_df['English'].transform(lambda x : 'START_ '+ x +
   ' _END')
   transformed_df['German'] =
   transformed_df['German'].transform(lambda x : 'START_ '+ x + '
   _END')

   return transformed_df
```

4.4 Text to Sequence Conversion

A Seq2Seq model requires that we convert both the input and the output sentences into integer sequences of fixed length.

4.5 Tokenization

For a neural network to predict on text data, it first must be turned into data it can understand. Since a neural network is a series of multiplication and addition operations, the input data needs to be number(s).For this project, each word and punctuation mark will be given a unique ID. (For other NLP projects, it might make sense to assign each character a unique ID.

When we run the tokenizer, it creates a word index, which is then used to convert each sentence to a vector.

```
def tokenize(sentences):
    # Create tokenizer
    text_tokenizer = Tokenizer()

    # Fit texts
    text_tokenizer.fit_on_texts(sentences)
    return text_tokenizer.texts_to_sequences(sentences), text_tokenizer
```

4.6 Padding

When batching the sequence of token'd words together, each sequence needs to be the same length. Since sentences are dynamic in length, we can add padding to the end of the sequences to make them the same length. Here we're going to use Kera's Pad_Sequences function.

```
eng_pad_sentence = pad_sequences(eng_tokenized, maxlen_eng, padding = "post")
ger_pad_sentence = pad_sequences(ger_tokenized, maxlen_ger, padding = "post")
```

5. Base Model Development

First, let’s break down the architecture of an RNN at a high level. Referring to the diagram above, there are a few parts of the model we need to be aware of:

5.1 Model Architecture

1. Inputs. Input sequences are fed into the model with one word for every time step. Each word is encoded as a unique integer or one-hot encoded vector that maps to the English dataset vocabulary.

2. Embedding Layers. 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). This is where 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. Outputs. The outputs are returned as a sequence of integers or one-hot encoded vectors which can then be mapped to the French dataset vocabulary.

5.2 Embeddings

Embeddings allow us to capture more precise syntactic and semantic word relationships. This is achieved by projecting each word into n-dimensional space. Words with similar meanings occupy similar regions of this space; the closer two words are, the more similar they are. And often the vectors between words represent useful relationships, such as gender, verb tense, or even geopolitical relationships. The dataset for this project has a small vocabulary and low syntactic variation, we'll use Keras to train the embeddings ourselves.

5.3 Encoder and Decoder

Our sequence-to-sequence model links two recurrent networks: an encoder and decoder. The encoder summarizes the input into a context variable, also called the state. This context is then decoded and the output sequence is generated.

Since both the encoder and decoder are recurrent, they have loops which process each part of the sequence at different time steps.

In our problem it takes four timesteps to encode the entire input sequence. At each time step, the encoder "reads" the input word and performs a transformation on its hidden state. Then it passes that hidden state to the next time step. Keep in mind that the hidden state represents the relevant context flowing through the network. The bigger the hidden state, the greater the learning capacity of the model, but also the greater the computation requirements. We'll talk more about the transformations within the hidden state when we cover gated recurrent units (GRU).

5.4 RNN Overview

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. It 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.

This is analogous to how we read. As we read, we are storing important pieces of information from previous words and sentences and using it as context to understand each new word and sentence.

Other types of neural networks can't do this (yet). Imagine we use a convolutional neural network (CNN) to perform object detection in a movie. Currently, there's no way for information from objects detected in previous scenes to inform the model's detection of objects in the current scene. For example, if a courtroom and judge were detected in a previous scene, that information could help correctly classify the judge's gavel in the current scene, instead of misclassifying it as a hammer or mallet. But CNNs don't allow this type of time-series context to flow through the network like RNNs do.

Depending on the use-case, we set up our RNN to handle inputs and outputs differently. For this project, we used a many-to-many process where the input is a sequence of English words and the output is a sequence of German words.

5.5 LSTM Overview

In Natural Language Processing (NLP), LSTMs are used for tasks such as text classification, sentiment analysis, language translation, and text generation. They are particularly useful for processing sequential data, such as text, where the order of words is important in determining the meaning of the text.

An LSTM network can be trained to predict the next word in a sentence given the previous words, or to classify the sentiment of a sentence based on its words and their order. The memory cells in LSTMs allow the network to capture long-term dependencies between words and their contextual relationships, helping it to better understand the meaning of a sentence.

LSTMs can be trained on large text corpora and fine-tuned on smaller NLP tasks, making them a popular choice in the NLP community. They are also often used in combination with other neural network models, such as Attention Mechanisms or Convolutional Neural Networks, to further improve performance.

5.6 Approach for This Project

To translate a corpus of German text to English or vice versa, we need to build a recurrent neural network (RNN) or long short term memory (LSTM) model.

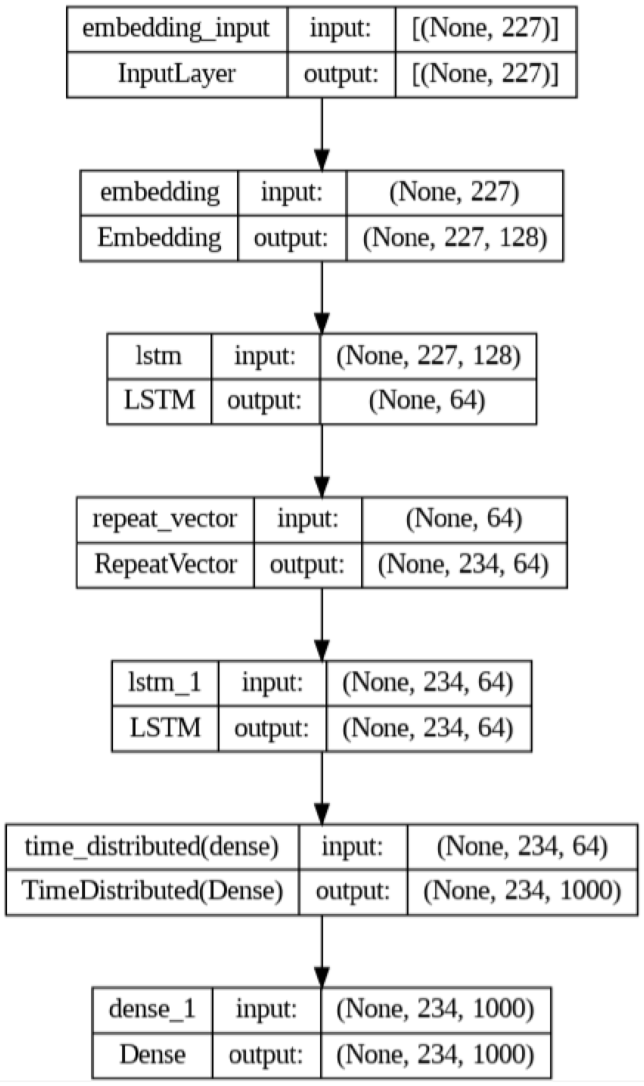
In this project we have taken an approach to experiment with several simple RNN and LSTM models. We have experimented with a total of 3 models.

1. Model 1: Basic LSTM model with full length of sentences to translate English sentences into German sentences.
2. Model 2: Basic RNN model with short length of sentences to translate German sentences into English sentences.
3. Model 3: Basic LSTM model with short length of sentences to translate German sentences

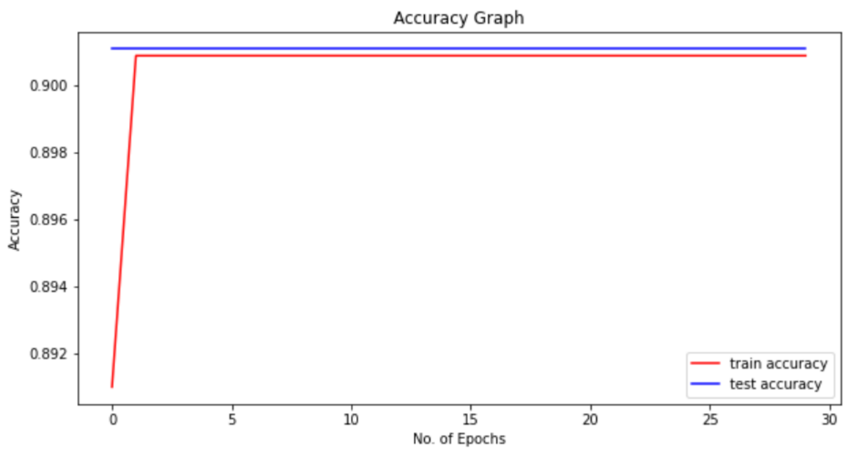
into English sentences.

5.6.1 Model 1 Summary and Performance

Model Summary:



Model Performance:



# Evaluating the model performance on test data

results_translate = model_translate_lstm.evaluate(X_test, Y_test)

```
175/175 [==============================] - 3s 15ms/step - loss: nan - accuracy:
0.9001
```

# Evaluating the model performance on train data

results_translate_train = model_translate_lstm.evaluate(X_train, Y_train)

```
524/524 [==============================] - 8s 15ms/step - loss: nan - accuracy:
0.9009
```

Model Predictions:

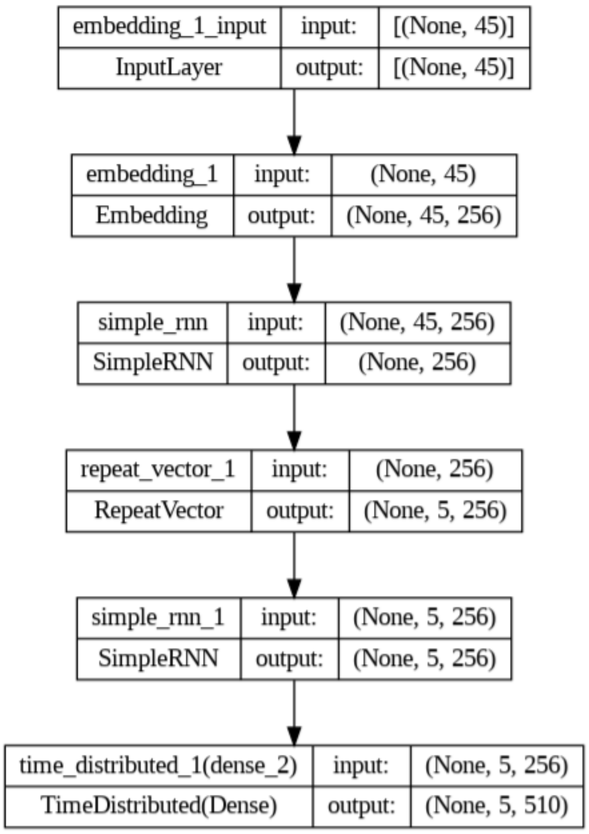
```
The english sentence is: START_ this house has already made a preliminary
assessment of what the subsequent costs will be _END
The german sentence is: START_ das parlament hat schon einen entwurf für die
folgekostenabschätzung verabschiedet _END
The predicted sentence is :
1/1 [==============================] - 1s 697ms/step
<empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty>
<empty> <empty> <empty> <empty> <empty> <empty> <empty>
```

Key points about the model:

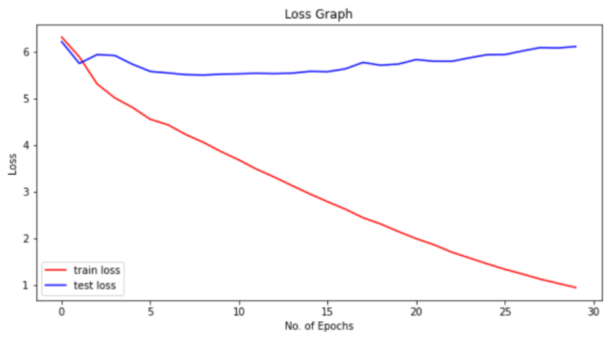
- Basic LSTM model is trained on 5% sample data. Due to the very large size of source data, unable to pre-process the entire dataset.
- Considering the large vocabulary size even with 5% sample data, the output node of the model is restricted to 1000.
- Model is generating similar accuracies on training and test datasets so model is not over fitting.
- Model accuracy is not improving after 2nd epoch and loss is 'nan' on both train and test datasets.
- Model is not predicting well and only generating a sequence of <empty> tokens as predicted sentences.
- So, we tried to observe model performance with shorter length of sentence with both LSTM and RNN methods.

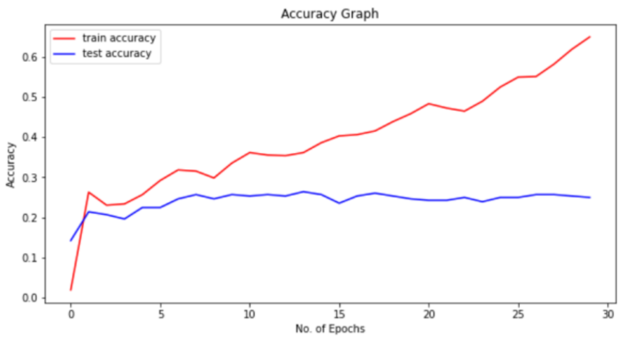
5.6.2 Model 2 Summary and Performance

Model Summary:


embedding_1_input | input: | [(None, 45)]
InputLayer | output: | [(None, 45)]
embedding_1 | input: | (None, 45)
Embedding | output: | (None, 45, 256)
simple_rnn | input: | (None, 45, 256)
SimpleRNN | output: | (None, 256)
repeat_vector_1 | input: | (None, 256)
RepeatVector | output: | (None, 5, 256)
simple_rnn_1 | input: | (None, 5, 256)
SimpleRNN | output: | (None, 5, 256)
time_distributed_1(dense_2) | input: | (None, 5, 256)
TimeDistributed(Dense) | output: | (None, 5, 510)

Model Performance:





```
# Evaluating the model performance
results_translate_rnn_test = model_translate_RNN.evaluate(testX, testY)

2/2 [==============================] - 0s 10ms/step - loss: 5.7033 - accuracy:
0.2381

# Evaluating the model performance on train data
results_translate_rnn_train = model_translate_RNN.evaluate(trainX, trainY)

6/6 [==============================] - 0s 8ms/step - loss: 2.8821 - accuracy:
0.5484
```

BLEU Score:

- BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high quality reference translations.

- The BLEU score compares a sentence against one or more reference sentences and tells how well the candidate sentence matched the list of reference sentences. It gives an output score between 0 and 1. A BLEU score of 1 means that the candidate sentence perfectly matches one of the reference sentences.

BLEU Score of Model 2:

```
BLEU-1: 0.081750
BLEU-2: 0.021242
BLEU-3: 0.000000
BLEU-4: 0.000000
```

Model Predictions:

```
Prediction of test sentences:
The German sentence is: das parlament wird sicherlich mithelfen
The English sentence is: parliament will certainly provide assistance
The predicted sentence is: that is that good good
**************************************************
The German sentence is: die turkei leugnet den armenischen holocaust
The English sentence is: turkey denies its armenian holocaust
The predicted sentence is: so is
**************************************************
The German sentence is: unser standpunkt ist vollkommen klar
The English sentence is: our position is absolutely clear
The predicted sentence is: we is is
**************************************************
The German sentence is: wie das funktionieren soll
The English sentence is: how
The predicted sentence is: they
**************************************************
The German sentence is: ruckwurfe von fischen
The English sentence is: fish discards
The predicted sentence is: welcome is
**************************************************
The German sentence is: bericht guinebertiere
The English sentence is: guinebertiere report
The predicted sentence is: european
**************************************************
```

Key points about the model:

- Basic RNN model is trained on 5% sample data. Due to the very large size of source data, unable to pre-process the entire dataset.

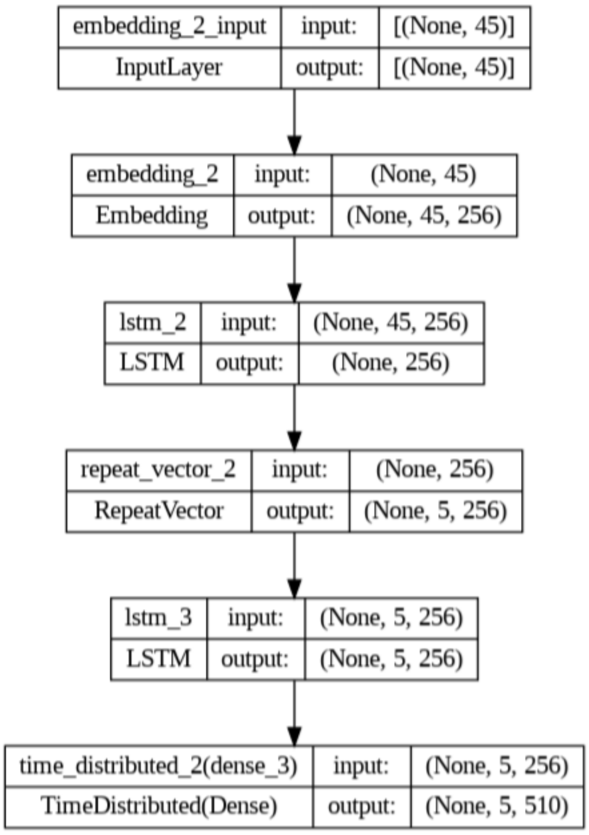
- In this model we tried an approach of forming the English-German sentence pairs and generated tokens from these pairs to train the model.

- This model is overfitting. Training data accuracy is high (54%) compared to test data accuracy (23%).

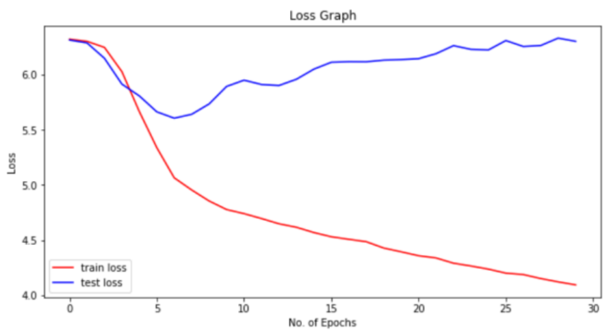
- Model accuracy on test dataset is staying in same range after 7th epoch, however it is contentiously increasing on train dataset.
- Model loss is sharply decreasing on train dataset, however it is slightly increasing after 7th epoch on test dataset.
- This model is predicting some words on test data, but it does not look like a generalized and stable model.

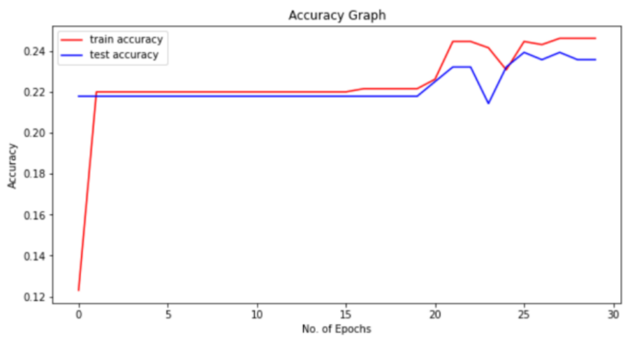
5.6.3 Model 3 Summary and Performance

Model Summary:


embedding_2_input | input: | [(None, 45)]
InputLayer | output: | [(None, 45)]
embedding_2 | input: | (None, 45)
Embedding | output: | (None, 45, 256)
lstm_2 | input: | (None, 45, 256)
LSTM | output: | (None, 256)
repeat_vector_2 | input: | (None, 256)
RepeatVector | output: | (None, 5, 256)
lstm_3 | input: | (None, 5, 256)
LSTM | output: | (None, 5, 256)
time_distributed_2(dense_3) | input: | (None, 5, 256)
TimeDistributed(Dense) | output: | (None, 5, 510)

Model Performance:





```
# Evaluating the model performance on test data

results_translate_lstm_test = model_translate_LSTM.evaluate(testX, testY)
2/2 [==============================] - 0s 7ms/step - loss: 5.9571 - accuracy:
0.2571

# Evaluating the model performance on train data

results_translate_lstm_train = model_translate_LSTM.evaluate(trainX, trainY)

6/6 [==============================] - 0s 8ms/step - loss: 4.7856 - accuracy:
0.2441
```

BLEU Score of Model 3:

```
BLEU-1: 0.056597
BLEU-2: 0.000000
BLEU-3: 0.000000
BLEU-4: 0.000000
```

Model Predictions:

```
Prediction of test sentences:
The German sentence is: das parlament wird sicherlich mithelfen
The English sentence is: parliament will certainly provide assistance
The predicted sentence is: is is
**************************************************
The German sentence is: die turkei leugnet den armenischen holocaust
The English sentence is: turkey denies its armenian holocaust
The predicted sentence is: is is
**************************************************
The German sentence is: unser standpunkt ist vollkommen klar
The English sentence is: our position is absolutely clear
The predicted sentence is: is is
**************************************************
The German sentence is: wie das funktionieren soll
The English sentence is: how
The predicted sentence is: is is
**************************************************
The German sentence is: ruckwurfe von fischen
The English sentence is: fish discards
The predicted sentence is: is

**************************************************
```

Key points about the model:

- Basic LSTM model is trained on 5% sample data. Due to the very large size of source data, unable to pre-process the entire dataset.
- In this model we tried an approach of forming the English-German sentence pairs and generated tokens from these pairs to train the model.
- This model is not overfitting. Training data accuracy and test data accuracy is almost similar (25%).

- Model accuracy on both test and train dataset is staying stable till 20th epoch then starts increasing.
- Model loss on train dataset is declining sharply, however loss on test dataset is decreasing up to 5th epoch then starts increasing.
- This model is predicting only some specific words eg. the, we, is, that.

6. Performance Improvement

In the next submission, we will implement the advanced models mentioned below to expect improved performance:

- Encoder_Decoder approach
- Bi-directional LSTM
- LSTM model with attention layer

7. References

1. Rico Sennrich, Biao Zhang "Revisiting Low-Resource Neural Machine Translation:A Case Study".:1905.11901
2. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the International Conference on Learning Representations (ICLR).
3. Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied Monolingual Data Improves Low-Resource Neural Machine Translation. In Proceedings of the Second Conference on Machine Translation, pages 148–156, Copenhagen, Denmark.
4. Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Unsupervised Statistical Machine Translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3632–3642, Brussels, Belgium.
5. Yun Chen, Yang Liu, Yong Cheng, and Victor O.K.Li. 2017. A Teacher-Student Framework for ZeroResource Neural Machine Translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1925–1935, Vancouver, Canada.
6. C¸aglar Gulc¸ehre, Orhan Firat, Kelvin Xu, Kyunghyun¨ Cho, Lo¨ıc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. CoRR, abs/1503.03535.
7. Barry Haddow, Nikolay Bogoychev, Denis Emelin, Ulrich Germann, Roman Grundkiewicz,

Kenneth Heafield, Antonio Valerio Miceli Barone, and Rico Sennrich. 2018. The University of Edinburgh’s Submissions to the WMT18 News Translation Task. In Proceedings of the Third Conference on Machine Translation, pages 403–413, Belgium, Brussels.

8. Toan Q. Nguyen and David Chiang. 2017. Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 296–301, Taipei, Taiwan.

9. Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 368–373, Berlin, Germany.

10. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, pages 3104–3112, Montreal, Quebec, Canada.

11. Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting Character-Based Neural Machine Translation with Capacity and Compression. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4295–4305, Brussels, Belgium

12. https://valueml.com/german-to-english-translator-using-keras-and-tensorflow-using-lstm-model/