Capstone Project – NLP Machine Translation

Group 09 - Final Report

Towards 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 resource's 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 "news-

commentary" 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.

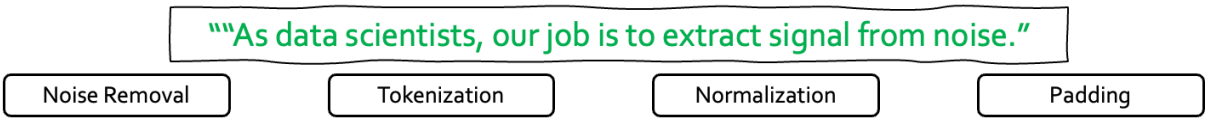
4. Overview of the Final Process

4.1 Data Preprocessing

As per the project requirement, we have used the pre-processed data from milestone 1 as an input to milestone 2 models. No pre-processing done in milestone 2.

In this section, we are explaining the approach taken in milestone 1 for the data cleaning 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.



4.2 Data Cleaning

4.2.1 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()

4.2.2 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()

4.2.3 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 the English language because it can impact the translation of English to German or vice versa.

```
# Function to pre_process data

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
```

4.2.4 Text to Sequence Conversion

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

4.2.5 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
```

4.2.6 Padding

When batching the sequence of tokened 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")
```

4.3 Algorithms Used in The Project

In this project we have taken an approach to experiment with several RNN and LSTM models. We have experimented with a total of 6 models.

1. Model 1: LSTM model with glove embedding to translate German sentences into English sentences.
2. Model 2: RNN model with glove embedding to translate German sentences into English sentences.
3. Model 3: Bidirectional LSTM model with glove embedding to translate German sentences into English sentences.
4. Model 4: Bidirectional RNN model with glove embedding to translate German sentences into English sentences.
5. Model 5: LSTM model with attention layer and glove embedding to translate German sentences into English sentences.
6. Model 6: Pre-trained MarianMT hugging face Helsinki-NLP/opus-mt-en-de model with transfer learning approach.

5. Step-by-Step Walk Through the Solution

First, let’s break down the architecture of an RNN and LSTM at a high level. 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 Embedding Technique Overview

Word2vec embeddings are based on training a shallow feedforward neural network while glove embeddings are learnt based on matrix factorization techniques.

Pre-trained word embeddings have proven to be invaluable for improving performance in natural language analysis tasks, which often suffer from paucity of data. Studies show that embeddings can be surprisingly effective in some cases – providing gains of up to 20 BLEU points in the most favorable setting. In word embeddings, every word is represented as an n-dimensional dense vector. The words that are similar will have similar vectors. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used.

"GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space."

For English sentences, i.e. the inputs, we will use the glove word embeddings. For the translated English sentences in the output, we will use custom word embeddings.

5.4 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.

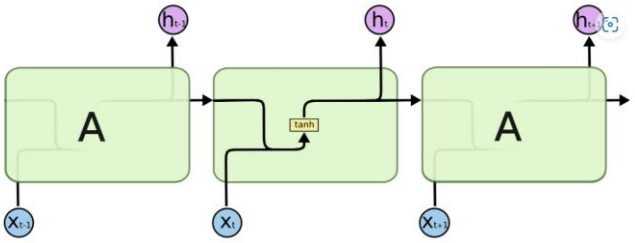
Since both the encoder and decoder are recurrent, they have loops which process each part of the sequence at different time steps.

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).

5.5 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.

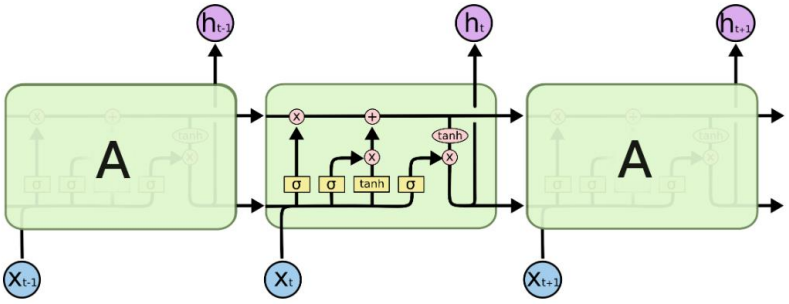
Depending on the use-case, we set up our RNN to handle inputs and outputs differently. For this project, we used a many-to-many process where the input is a sequence of English words, and the

output is a sequence of German words.

5.6 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.

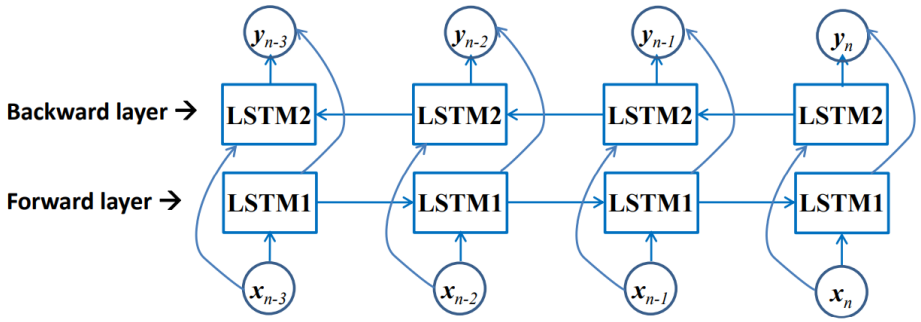
An LSTM network can be trained to predict the next word in a sentence given the previous words, or to classify the sentiment of a sentence based on its words and their order. The memory cells in LSTMs allow the network to capture long-term dependencies between words and their contextual relationships,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



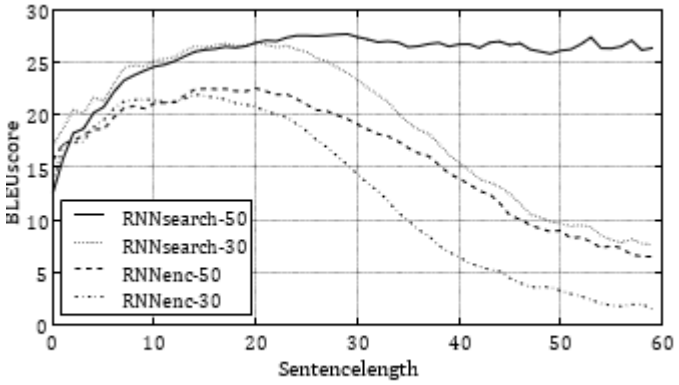
LSTMs can be trained on large text corpora and fine-tuned on smaller NLP tasks, making them a popular choice in the NLP community. They are also often used in combination with other neural network models, such as Attention Mechanisms or Convolutional Neural Networks, to further improve performance.

5.7 Bidirectional Model Overview

Here we employed sequence to sequence architecture (seq2seq) with Bahdanau's Attention mechanism for NMT. As we have mentioned earlier the problem with the Encoder - Decoder approach is that it calculates a word vector outputted from the last LSTM unit of the encoder section which is further carried on to the decoder section for translation. The problem is evident that the word vector which is calculated only at the end of the input string may not be able to carry the entire information or the context of the entire sentence or part of the sentence. Now this is like a human translator who listens to an entire sentence or a group of sentences and then starts translating. It is the choice of that person how long he listens and starts translation. The problem with this context vector is solved using, Attention mechanism adopted by Bahdanau and his fellow researchers in the conference paper presented at ICLR 2015.



The usual problem in RNN/LSTM is that we only have forward layers so they will not have information of next sequence words, therefore without proper context the sentence model might not predict the right words. The Bleu score chart plotted against length of the sentences seem to be drastically decreasing thus decreasing the prediction quality. As seen from the figure below the BLEU score is lowering as the length of input sentences increases. In the case of Bidirectional we have a forward and backward layer with that model that can have information of both previous and next words, therefore with proper context the sentence model will predict better.



The figure is as is from the paper under study.

5.8 Attention Layer Overview

Why use Attention?

To predict words, some of the previous or next words are important. As we talked in Bidirectional discussion. But which word is more important. To find that we use Attention model to give importance to words. Then the model can concentrate on one word which has more importance.

Here is how attention works: -

After getting output Z (in attention image) which is a concatenation of forward and backward hidden states [h, h`] the first step is to calculate attention weights(α).

To calculate α we need score(e) which is calculated using the above formula. The score is based on decoder's (LSTM) hidden state(before predicting y) and output of encoder Z.

Then the context vector (Ct) is computed as a weighted sum of attention weights(α) and output of encoder(Z).

Output y is generated by the decoder using context vector (Ct) as input.

From this paper by google https://arxiv.org/pdf/1609.08144.pdf which explains mathematical formulation of the encoder-decoder models

Conditional probability for translation

Here Y is the translated word(German), X is the given word(English) and x_i are the hidden state output of the encoder for input word X. The above equation says that the probability of translated y_i is conditioned upon previously translated words (y0 to y_i-1) and hidden state output x_i for the encoder.

Prepare Data for Model: -

Add SOS (Start Of String) and EOS(End Of String) tokens in sentences of target language. Due to these tokens, we can have length of target sentences and input sentences different from each other. And it also helps decoders to start and stop predicting.

Note: - length of inputs (i.e. English sentences) need to be same and and length of target (i.e. German) should be the same but they can be different from each other.

Tokenize: - Neural networks do not accept text as input so we'll have to convert them into numbers. To do so we will use Tensorflow's Tokenizer.

This tokenizer is very helpful from which we can get frequency of words, dictionaries of word to index & index to word. Which will be used to convert words into numbers (for training) and numbers into words(for prediction)

Padding: - Neural networks also need input (i.e. sentences) in same length So we'll pad sentences of English and Marathi language with '0' to get length of sentences as maximum length sentence of respective language.

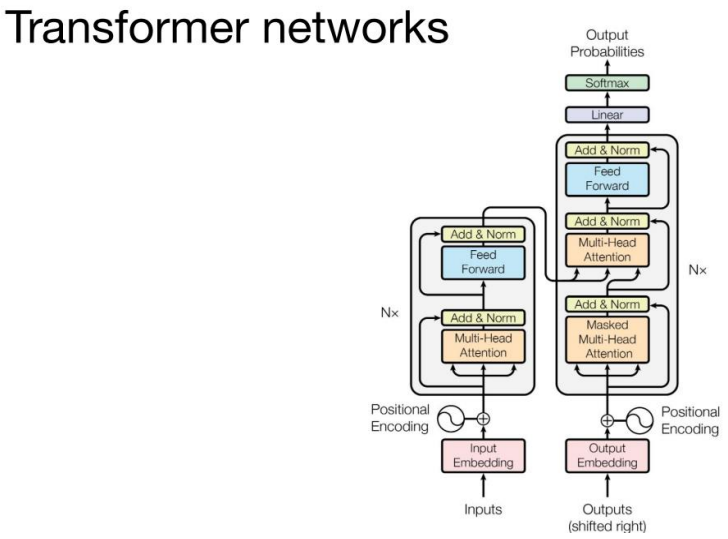
Encoder: As we discussed previously, we will use Bidirectional LSTM in encoder. It will learn patterns in input language (i.e English). We will use both the encoder's outputs and its states(context vector[h, c]). Now taking states is little different in Bidirectional because it has forward and backward states so we will have to consider both. (i.e. We will concatenate them.)

Decoder: In decoder we use simple LSTM. Attention layer code was borrowed code for attention from BahdanauAttention.py file.

Inference Model: -

We use this model to predict output sequences by using weights of pre-trained models. Here we cannot just apply model.predict() as other ML and DL models because in our case encoder model

learns features in input sentences and decoder simply takes encoder states and predicts word by word using decoder inputs. So, for prediction we have to do the same process.



5.9 Transfer Learning Overview

Transfer learning is a technique in which a deep learning model trained on a large dataset is used to perform similar tasks on another dataset. These deep learning models are called pre-trained models. Breakthrough of transfer learning firstly occurred in computer vision in the year 2012-13. However, with recent advances in NLP, transfer learning has become a viable option in NLP as well.

These pre-trained models do not process an input sequence token by token rather they take the entire sequence as input in one go which is a big improvement over RNN based models because now the model can be accelerated by the GPUs.

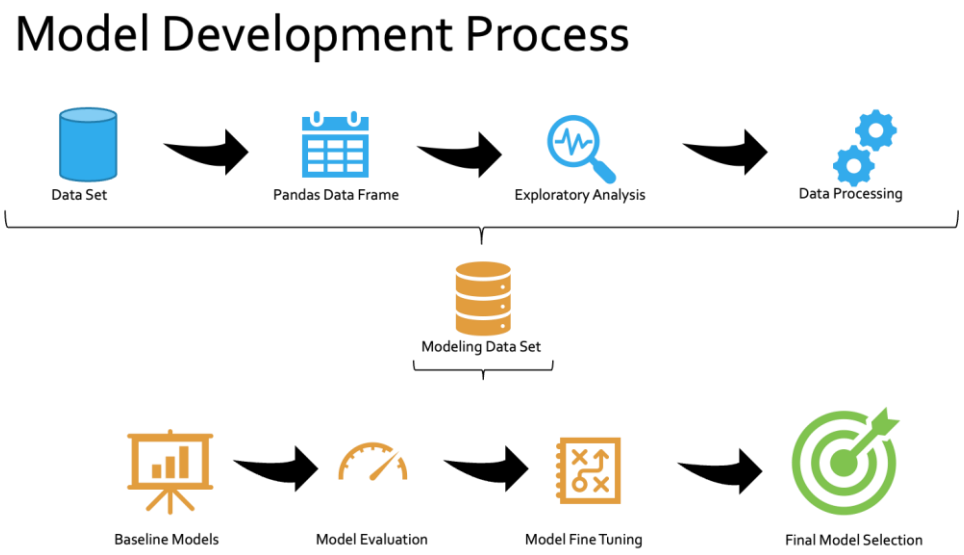
There is no need for labeled data to pre-train these models. It means that we must just provide a huge amount of unlabeled text data to train a transformer-based model. This is how transfer learning works in NLP.

Transfer Learning Techniques:

- Train the entire architecture – We can further train the entire pre-trained model on our dataset and feed the output to a SoftMax layer. In this case, the error is backpropagated through the entire architecture and the pre-trained weights of the model are updated based on the new dataset.
- Train some layers while freezing others – Another way to use a pre-trained model is to train it partially. What we can do is keep the weights of initial layers of the model frozen while we retrain only the higher layers. We can try and test as to how many layers to be frozen and how many to be trained.
- Freeze the entire architecture – We can even freeze all the layers of the model and attach a few neural network layers of our own and train this new model. Note that the weights of only the attached layers will be updated during model training.

5.10 Approach for This Project

As it is apparent, the problem we are dealing with involves classification of text-based input into pre-defined groups, we follow the following standard approach to arrive at the solutions –



- Understanding of the existing process
- Data Acquisition
- Exploratory Data Analysis
- Design of the solution
- Data pre-processing
- Model training
    - LSTM with glove embeddings
    - RNN with glove embeddings
    - Bidirectional LSTM
    - Bidirectional RNN
    - LSTM with attention layer
    - Transfer Learning
- Model Fine Tuning

- Final Model Selection
- Comparison of Base-line vs Final Model
- Conclusion

Based on the observations from the basic RNN and LSTM models implemented in milestone 1, we have taken data with both German and English sentences up to length of 30 words as models were not performing well with full length sentences. Also, computationally it is very expensive to train the models with full length sentences. As per the requirement of milestone 2, data pre-processed in milestone 1 is taken as an input for the models implemented in milestone 2. We observed that newly developed models are not translating all the words and sentences properly so we also tried the transfer learning approach with pre-trained models from hugging face library. In the transfer learning approach we have used the original full length sentences to check the output.

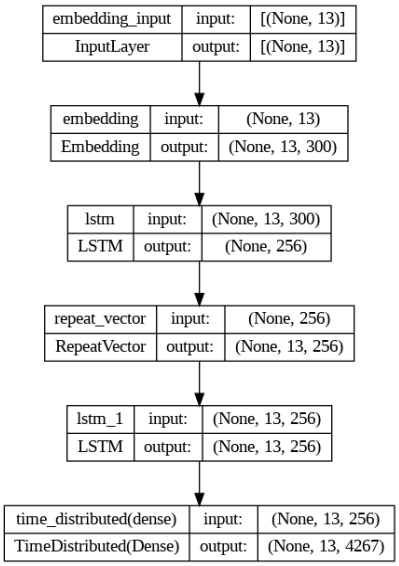
6. Model Evaluation

BLEU Score:

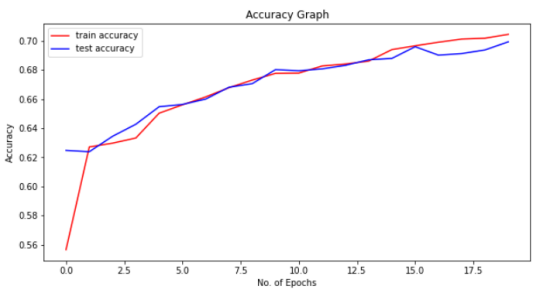
- BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations.
- The BLEU score compares a sentence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

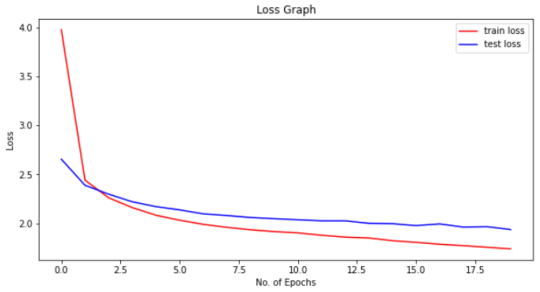
6.1 Model 1: LSTM with Glove Embeddings

Model Summary:



Model Performance:






```
# Evaluating the model performance on test data

results_translate_lstm_test = model_lstm_embeddings.evaluate(testX_embed, testY_embed)
```

```
75/75 [==============================] - 1s 8ms/step - loss: 1.9024 - accuracy: 0.7036
```

```
# Evaluating the model performance on train data

results_translate_lstm_train = model_lstm_embeddings.evaluate(trainX_embed, trainY_embed)
```

```
225/225 [==============================] - 2s 7ms/step - loss: 1.7731 - accuracy: 0.7048
```

BLEU Score (Test Data):

```
BLEU-1: 0.449680
```

Model Predictions:

```
Prediction of test sentences:
The German sentence is: start das ist das falsche signal end
The English sentence is: start that is the wrong signal end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start das ist sehr wichtig end
The English sentence is: start this is vital end
The predicted sentence is: start that is not end end
**************************************************
The German sentence is: start warum brauchen wir ihn end
The English sentence is: start why do we need it end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start ich habe da groe hoffnung end
The English sentence is: start i have high hopes of them end
The predicted sentence is: start that is is end end end
**************************************************
The German sentence is: start gesundheitsberichterstattung end
The English sentence is: start health monitoring end
The predicted sentence is: start report end
**************************************************
```

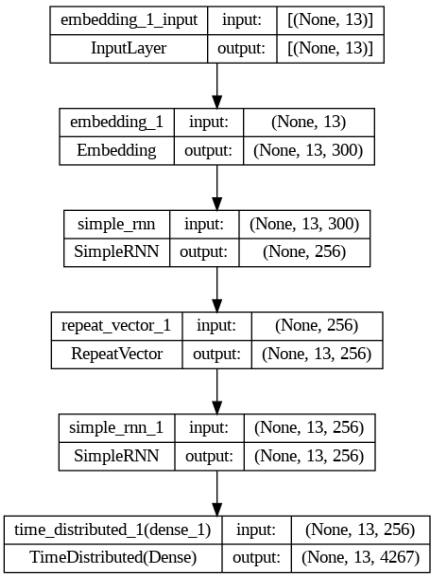
Key points about the model:

- Model is looking balanced and generalized with both training and test data accuracy of 70%. This model is not overfitting.
- Model has BLEU score of 0.44 on both training and test dataset.
- Both training and test accuracy has seen peaks and valleys, however it has increased with increase in number of epochs.

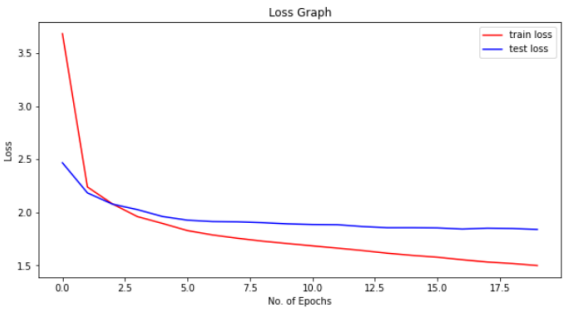
- Training loss has decreased sharply at 2nd epoch and thereafter decreased slowly with increase in number of epochs.
- Test loss has decreased slowly with increase in number of epochs.
- Model is translating German sentences into English, but it is translating only certain words.

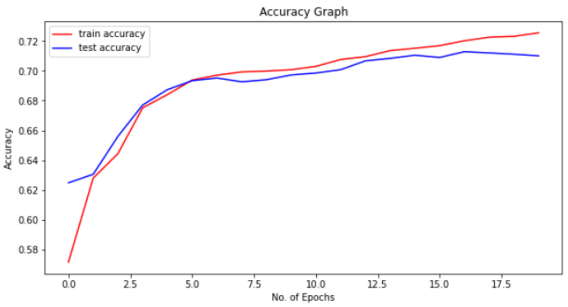
6.2 Model 2: RNN with Glove Embeddings

Model Summary:



Model Performance:






```
# Evaluating the model performance on test data

results_translate_rnn_test = model_rnn_embeddings.evaluate(testX_embed, testY_embed)
```

```
75/75 [==============================] - 1s 9ms/step - loss: 1.8088 - accuracy: 0.7147
```

```
# Evaluating the model performance on train data

results_translate_rnn_train = model_rnn_embeddings.evaluate(trainX_embed, trainY_embed)
```

```
225/225 [==============================] - 2s 7ms/step - loss: 1.5659 - accuracy: 0.7244
```

BLEU Score (Test Data):

```
BLEU-1: 0.490837
```

Model Predictions:

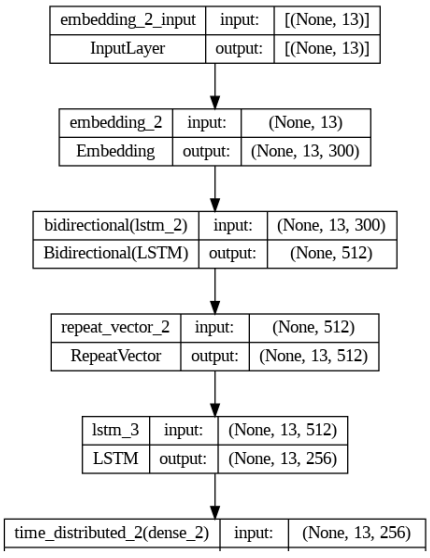
```
Prediction of test sentences:
The German sentence is: start das ist das falsche signal end
The English sentence is: start that is the wrong signal end
The predicted sentence is: start that is a the end end
**************************************************
The German sentence is: start das ist sehr wichtig end
The English sentence is: start this is vital end
The predicted sentence is: start that is a end end
**************************************************
The German sentence is: start warum brauchen wir ihn end
The English sentence is: start why do we need it end
The predicted sentence is: start i are we end end
**************************************************
The German sentence is: start ich habe da groe hoffnung end
The English sentence is: start i have high hopes of them end
The predicted sentence is: start we is we end end
**************************************************
The German sentence is: start gesundheitsberichterstattung end
The English sentence is: start health monitoring end
The predicted sentence is: start end end
**************************************************
```

Key points about the model:

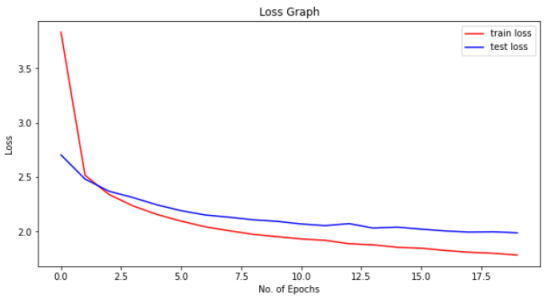
- RNN model with glove embeddings is looking balanced and generalized model with both training and test data accuracy close to 72%. This model is also not overfitting.
- RNN model with glove embedding has BLEU score of 0.49 on both training and test dataset.
- Both training and test accuracy has sharply increased till 5th epoch and thereafter increased slowly with increase in number of epochs.
- Training loss has decreased sharply at 2nd epoch and thereafter decreased slowly with increase in number of epochs.
- Test loss has decreased till 5th epoch thereafter not much reduction.
- Model is translating German sentences into English, but it is translating only certain words.

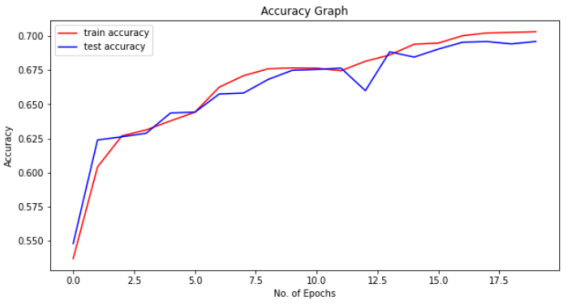
6.3 Model 3: Bidirectional LSTM

Model Summary:



Model Performance:






```
# Evaluating the model performance on test data

results_translate_lstmbi_test = model_bidirectionallstm_embeddings.evaluate(testX_embed, testY_embed)
```

```
75/75 [==============================] - 1s 9ms/step - loss: 1.9523 - accuracy: 0.7013
```

```
# Evaluating the model performance on train data

results_translate_lstmbi_train = model_bidirectionallstm_embeddings.evaluate(trainX_embed, trainY_embed)
```

```
225/225 [==============================] - 2s 8ms/step - loss: 1.8313 - accuracy: 0.7002
```

BLEU Score (Test Data):

```
BLEU-1: 0.438930
```

Model Predictions:

```
Prediction of test sentences:
The German sentence is: start das ist das falsche signal end
The English sentence is: start that is the wrong signal end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start das ist sehr wichtig end
The English sentence is: start this is vital end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start warum brauchen wir ihn end
The English sentence is: start why do we need it end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start ich habe da groe hoffnung end
The English sentence is: start i have high hopes of them end
The predicted sentence is: start that is is end end
**************************************************
The German sentence is: start gesundheitsberichterstattung end
The English sentence is: start health monitoring end
The predicted sentence is: start end end
**************************************************
```

Key points about the model:

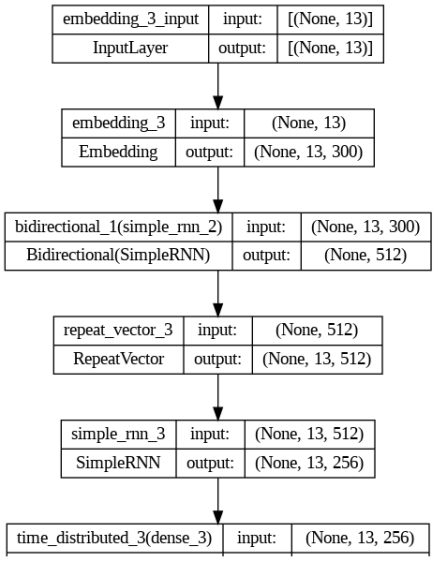
- Model is looking balanced and generalized model with both training and test data accuracy of 70%. This model is not overfitting.
- Model has BLEU score of 0.43 on both training and test dataset.
- Training accuracy has increased till 7th epoch and thereafter increased slowly with increase in number of epochs.
- Test accuracy has increased till 3rd epoch and thereafter increased slowly with increase in

number of epochs. However, there are peaks and valleys in test accuracy after 3rd epoch.

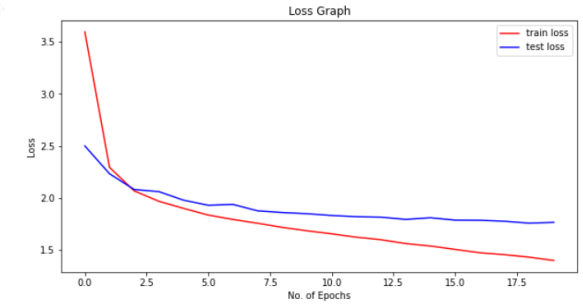
- Training loss has decreased sharply at 2nd epoch and thereafter decreased slowly with increase in number of epochs.
- Test loss has decreased slowly with increase in number of epochs.
- Model is translating German sentences into English, but it is translating only certain words.

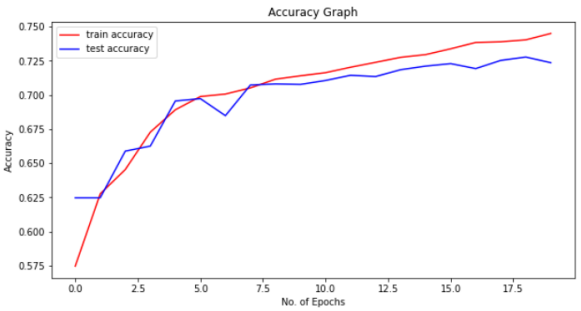
6.4 Model 4: Bidirectional RNN

Model Summary:



Model Performance:






```
# Evaluating the model performance on test data

results_translate_rnnbi_test = model_bidirectionalrnn_embeddings.evaluate(testX_embed, testY_embed)
```

```
75/75 [==============================] - 1s 9ms/step - loss: 1.7261 - accuracy: 0.7283
```

```
# Evaluating the model performance on train data

results_translate_rnnbi_train = model_bidirectionalrnn_embeddings.evaluate(trainX_embed, trainY_embed)
```

```
225/225 [==============================] - 2s 9ms/step - loss: 1.4778 - accuracy: 0.7412
```

BLEU Score (Test Data):

```
BLEU-1: 0.525460
```

Model Predictions:

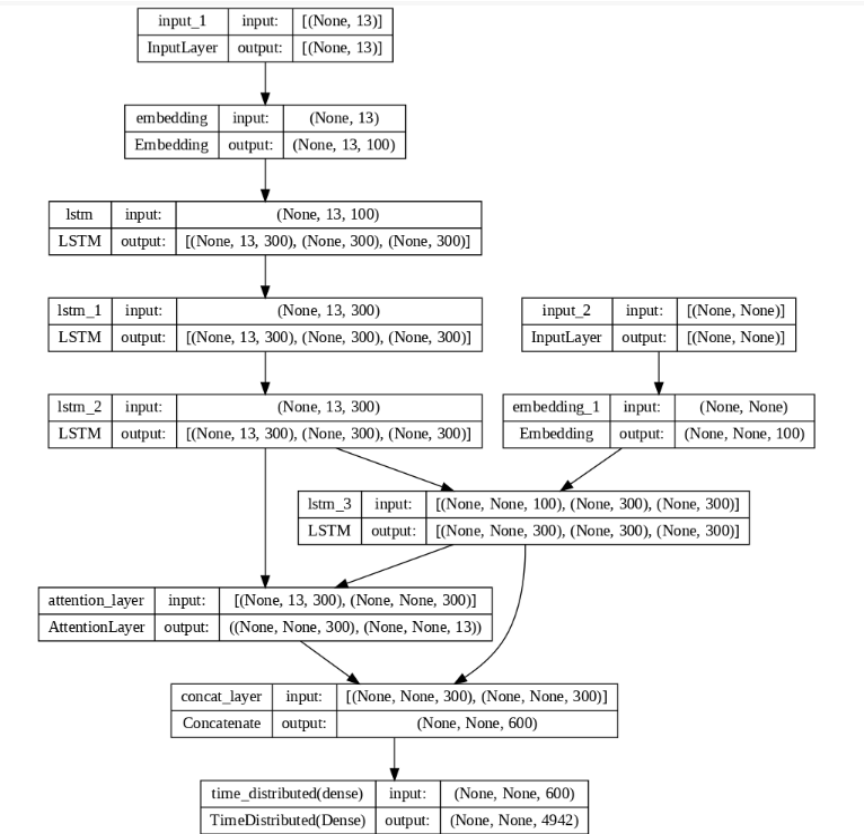
```
Prediction of test sentences:
The German sentence is: start das ist das falsche signal end
The English sentence is: start that is the wrong signal end
The predicted sentence is: start that is a a end end
**************************************************
The German sentence is: start das ist sehr wichtig end
The English sentence is: start this is vital end
The predicted sentence is: start that is a end end
**************************************************
The German sentence is: start warum brauchen wir ihn end
The English sentence is: start why do we need it end
The predicted sentence is: start we will we that end end
**************************************************
The German sentence is: start ich habe da groe hoffnung end
The English sentence is: start i have high hopes of them end
The predicted sentence is: start i is not it end
**************************************************
The German sentence is: start gesundheitsberichterstattung end
The English sentence is: start health monitoring end
The predicted sentence is: start applause end
**************************************************
```

Key points about the model:

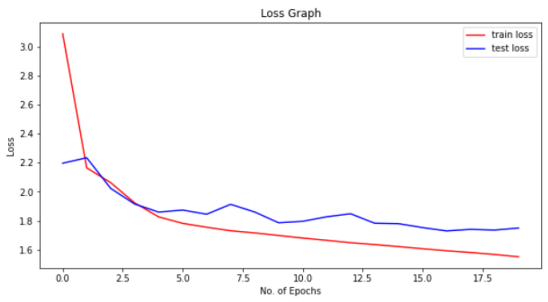
- Model is looking balanced and generalized model with training accuracy just over 74% and test accuracy close to 73%. This model is also not overfitting.
- Model has BLEU score close to 0.53 on both training and test dataset.
- Both training and test accuracy has sharply increased till 5th epoch and thereafter increased slowly with increase in number of epochs. Test accuracy has seen some peaks and valleys after 5th epoch.
- Training loss has decreased sharply at 2nd epoch and thereafter decreased slowly with increase in number of epochs.
- Test loss has decreased sharply till 3rd epoch thereafter decreased slowly with increase in number of epochs.
- Model is translating German sentences into English. Model is translating more words compared to RNN with glove embedding model.

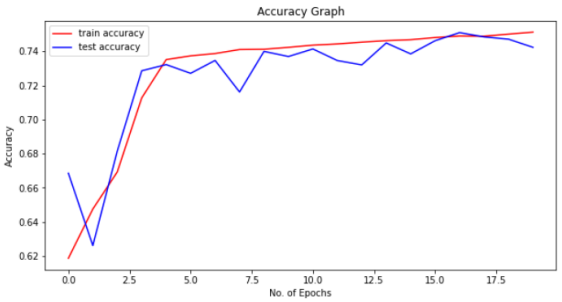
6.5 Model 5: LSTM with Attention

Model Summary:



Model Performance:






```
model.evaluate([X_test,Y_test[:,:-1]],Y_test.reshape(Y_test.shape[0],Y_test.shape[1],1)[:,1:])
```

```
75/75 [==============================] - 1s 19ms/step - loss: 1.7503 - accuracy: 0.7423
[1.7503184080123901, 0.7423103451728821]
```

```
model.evaluate([X_train,Y_train[:,:-1]],Y_train.reshape(Y_train.shape[0],Y_train.shape[1],1)[:,1:])
```

```
225/225 [==============================] - 4s 19ms/step - loss: 1.5800 - accuracy: 0.7422
[1.5800246000289917, 0.7422027587890625]
```

Model Predictions:

```
English Sentence is: start there is no abracadabra end
German Sentence is: es gibt kein abrakadabra
1/1 [==============================] - 1s 711ms/step
1/1 [==============================] - 0s 424ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
Predicted Sentence is:  das ist ich nicht
**************************************************
English Sentence is: start we were deceived end
German Sentence is: wir wurden betrogen
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
Predicted Sentence is:  das ist nicht nicht
**************************************************
English Sentence is: start valid votes end
German Sentence is: gultige stimmen
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
Predicted Sentence is:  votes views com
**************************************************
English Sentence is: start we need to get them through end
German Sentence is: wir mussen sie verabschieden
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
Predicted Sentence is:  das ist nicht nicht
**************************************************
English Sentence is: start what does that suggest end
German Sentence is: wo sind wir denn hier
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
Predicted Sentence is:  das ist ich nicht
**************************************************
```

BLEU Score (Test Data):

```
BLEU-1: 0.000000
```

Key points about the model:

- Model is looking balanced and generalized with both training and test accuracy of 75%. This model is also not overfitting.
- Model has BLEU score of 0.00 on test dataset.
- Training accuracy has sharply increased up to 4th epoch and thereafter increased slowly with increase in number of epochs.
- Test accuracy has decreased after 1st epoch, however increased sharply till 3rd epoch thereafter it has increased with some peaks and valleys.

- Training loss has decreased sharply at 1st and 2nd epoch and thereafter decreased slowly with increase in number of epoch's.
- Test loss has slightly increased after 1st epoch then decreased till 4th epoch. Thereafter it has observed peaks and valleys and slowly decreased with increase in number of epochs.
- Model is translating German sentences into English. Model is translating more words compared to Bidirectional LSTM with glove embedding model.

6.6 Model 6: Transfer Learning with Pre-Trained Hugging Face Helsinki-NLP/opus-mt-en-de Model

We have used the MarianMT translation pre-trained model which is developed by the Language Technology Research Group at the University of Helsinki and is trained on OPUS data.

We have used this model directly on original full-length sentences to check the output, hence we have not generated performance evaluation metrics.

Model Predictions:

Key points about the model:

- Model MarianMT is giving excellent performance with accurately translating all the sentences.
- Model is translating English sentences into German.
- Model is translating original full-length sentences from the dataset.

7. Comparison to Benchmark

Model	Train Accuracy	Test Accuracy
LSTM with Glove Embeddings	70.48%	70.36%
RNN with Glove Embeddings	72.44%	71.47%

Bidirectional LSTM with Glove Embeddings	70.02%	70.13%
Bidirectional RNN with Glove Embeddings	74.12%	72.83%
LSTM with Attention Layer	74.22%	74.23%

For Transfer learning models we have not measured the performance. We have used them directly on the original data set with full length sentences to check the output.

Based on the performances of the basic RNN and LSTM models implemented in milestone 1, we have set up a benchmark to improve the model accuracy beyond 64%. In all the 5 models implemented in milestone 2 we have managed to achieve the benchmark.

We also have set up a benchmark to reduce the model overfitting and make them stable and generalized. In all the 5 models implemented in milestone 2 we have managed to achieve this.

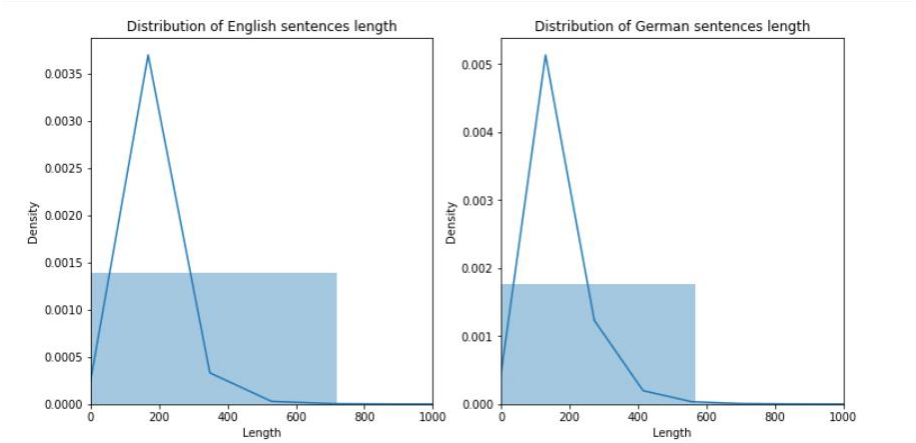
In the benchmark models from milestone 1 we had used the short length sentences up to sentence length of 6 words. In milestone 2, we have increased the sentence length up to 30 words and are able to achieve better translation results compared to basic models from milestone 1.

8. Visualizations

8.1 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

8.2 Sentence Distribution



9. Implications

- Improved performance in NLP tasks: LSTMs are known to be effective at modeling sequential data, making them well-suited for tasks that involve analyzing natural language data. By capturing long-term dependencies in text, LSTMs can often achieve better performance than other types of models.

- More complex models: LSTMs are a more complex type of model than other types of neural networks, which means that they can require more computational resources and take longer to train.

- Transfer learning: LSTM models can be pre-trained on large datasets to learn general features of text data, and then fine-tuned on smaller datasets for specific tasks. This approach, known as transfer learning, can save time and resources and often leads to better performance.

- Interpreting model outputs: Because LSTMs are a black-box model, it can be challenging to interpret their outputs and understand how they are making decisions. This can be a drawback in applications where interpretability is important, such as healthcare or finance.

10. Limitations

- Difficulty with long-term dependencies: LSTMs can struggle to capture dependencies between words that are far apart in a sentence, which can limit their ability to understand complex relationships in text.

- Memory and computational requirements: LSTMs can require a lot of memory and computational resources, which can make them slow or even impractical for processing very large datasets.

- Overfitting: LSTMs can sometimes be overfit to the training data, which means they may

not generalize well to new data.

- Difficulty with rare words: LSTMs can have trouble handling rare words or words that are not in their vocabulary, which can limit their effectiveness in some applications.

11. Closing Reflections

- Considering the performances of all the implemented 5 models, we observe that Bi-directionals RNN with glove embedding model is having better accuracy and stable performance.
- This model is translating more sentences accurately compared to other models.
- Hence, we consider this model as the best performing model.
- Looking at the translation accuracy and performance of all the models, we can conclude that pre-trained models are much better as they are trained on huge corpus. They are translating full length sentences accurately.
- Newly developed models from scratch do not work well on full length sentences and with short length sentences their translation performance is not that good.
- We have also observed that building new translation models from scratch is computationally very expensive. With limited resources we are not able to train them on large datasets. So, it is better to use a transfer learning approach with pre-trained models for translation use cases instead of developing new models from scratch.

12. 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/

13. https://huggingface.co/Helsinki-NLP/opus-mt-de-en

14. https://huggingface.co/google/bert2bert_L-24_wmt_de_en

15. https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/