Machine Learning Engineer Nanodegree

Capstone Project

Sanya Saxena

23 June 2018

I. Definition: German to English Translator

This project deals with translation of German words to English.

Project Overview:

One of the earliest goals for computers was the automatic translation of text from one language to another. Automatic or machine translation is perhaps one of the most challenging artificial intelligence tasks given the fluidity of human language. Classically, rule-based systems were used for this task, which were replaced in the 1990s with statistical methods. More recently, deep neural network models achieve state-of-the-art results in a field that is aptly named neural machine translation. Machine translation is a challenging task that traditionally involves large statistical models developed using highly sophisticated linguistic knowledge. Neural machine translation is the use of deep neural networks for the problem of machine translation. Sequence prediction often involves forecasting the next value in a real valued sequence or outputting a class label for an input sequence. This is often framed as a sequence of one input time step to one output time step (e.g. one-to-one) or multiple input time steps to one output time step (many-to-one) type sequence prediction problem.

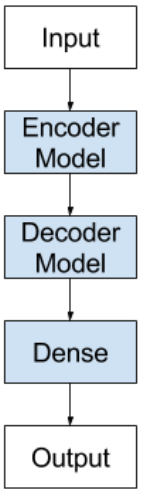
In this project, I have tried to translate German sentences to corresponding English translations. This translation process makes use of Neural Machine Translation with the help of sequence to sequence model with the help of external dataset, which is used in Tatoeba project. This dataset is set of English to German word pairs.

Problem Statement:

In this project, we are trying to build a German to English translator with the concept of Neural Machine Translation. The project will take German words as inputs and translate the given words to English.
In general it takes up a small problem of the larger domain of language translation problem. The Neural Machine Translation approach that is taken up in this project is with the help of Keras (Sequence to Sequence Model). This successfully takes any one language as input and translates it desired language. In this project, we deal with German to English Translation.
The approach of the project is mentioned below:

1. The dataset containing English-German translated word pair is loaded.
2. Cleaning of data
3. Model is trained in Keras by sequence to sequence Model with help of LSTM Networks.
4. The output is tested with the testing set and BLEU is calculated to check the performance of the model.

If the model translates the input, we encounter success. We will see the input words, their correct English translation and the English prediction by our model. The model is tested by the BLEU score. The LSTM architecture is shown below:



Metrics:

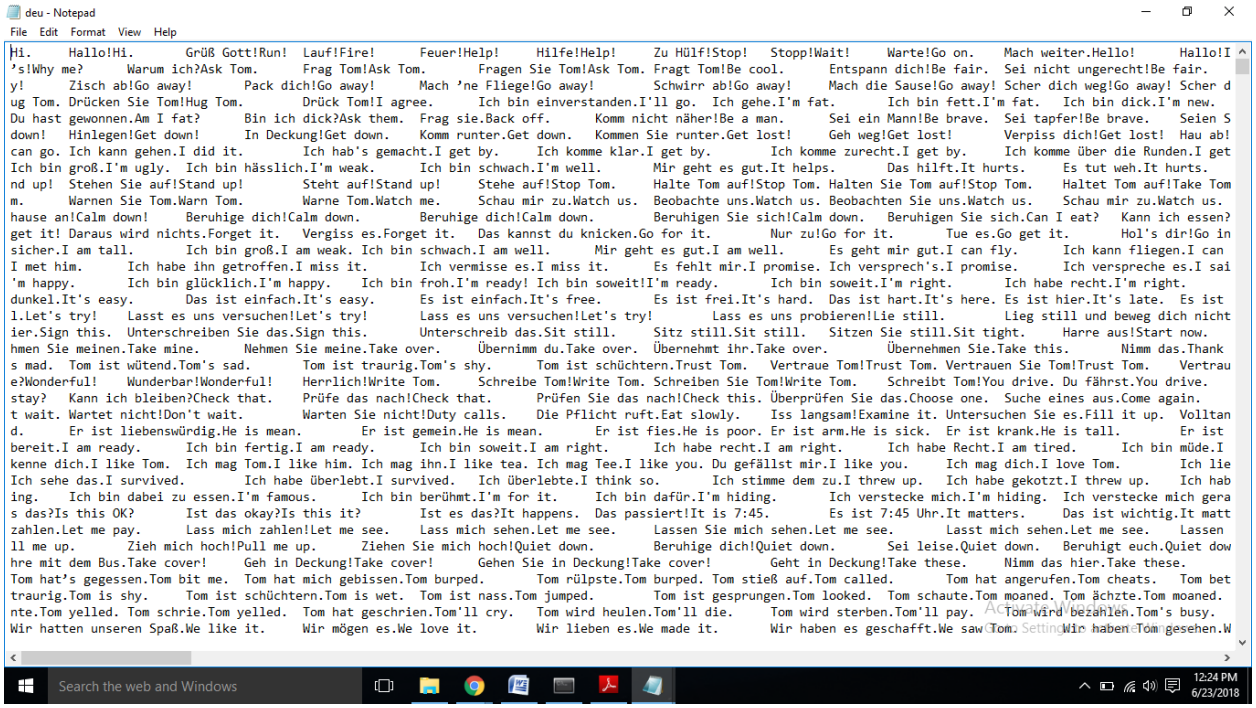
The project can be evaluated on basis of BLEU score. **BLEU** (**bilingual evaluation understudy**) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU.
BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts.

Since this metric evaluates the performance of Machine Translations, it would a nice option for evaluation in this project which deals with Machine Translation at its core.

II. Analysis

Data Exploration:

We will use a dataset of German to English terms used as the basis for flashcards for language learning.
The dataset is available from the ManyThings.org website, with examples drawn from the Tatoeba Project. The dataset is comprised of German phrases and their English counterparts and is intended to be used with the Anki flashcard software.
The dataset can be downloaded from:
http://www.manythings.org/anki/
The page provides a list of many language pairs. We have a file called *deu.txt* that contains 152,820 pairs of English to German phases, one pair per line with a tab separating the language.
We will frame the prediction problem as given a sequence of words in German as input, translate or predict the sequence of words in English.



Above is a picture of the dataset that is used. We can see that the English-German words pairs are present. However there are duplicates present. Also the file is not formatted. We have to separate these pairs and disintegrate these long sentences. The punctuations and Uppercases can cause anomalies. Hence data cleaning would have to be done before training model.

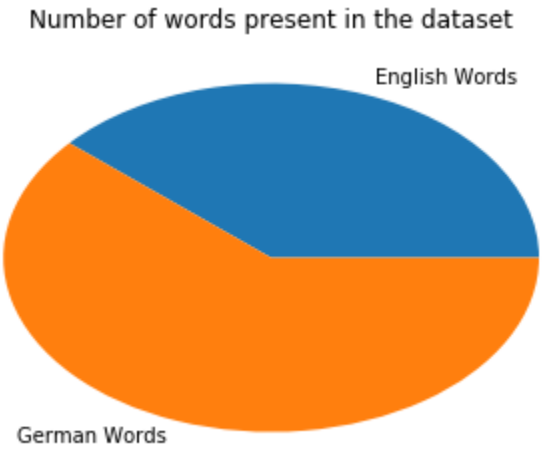
Exploratory Visualization:

After proper cleaning of data, we obtain pairs of English to German words in each line. This cleaned data is stored in file named english-german.pkl. This is evident from below code.

```
# load dataset
filename = 'deu.txt'
doc = load_doc(filename)
# split into english-german pairs
pairs = to_pairs(doc)
# clean sentences
clean_pairs = clean_pairs(pairs)
# save clean pairs to file
save_clean_data(clean_pairs, 'english-german.pkl')
# spot check
for i in range(10):
    print('[%s] => [%s]' % (clean_pairs[i,0], clean_pairs[i,1]))
```

```
Saved: english-german.pkl
[hi] => [hallo]
[hi] => [gru gott]
[run] => [lauf]
[fire] => [feuer]
[help] => [hilfe]
[help] => [zu hulf]
[stop] => [stopp]
[wait] => [warte]
[go on] => [mach weiter]
[hello] => [hallo]
```

Now we further analyze these words that are available with us. We check for the number of English words and German words. We plot a pie chart for the same.



It is evident from the above figure that the number of German words is more than the number of English words. When we look for those numbers, we find that English words are 2315 while there are 3686 German words.

It implies that there are some English words which have more than single translations in German. Inversely, we can also say that there are some German words which point to the same English translation. To be specific the count of such words is:

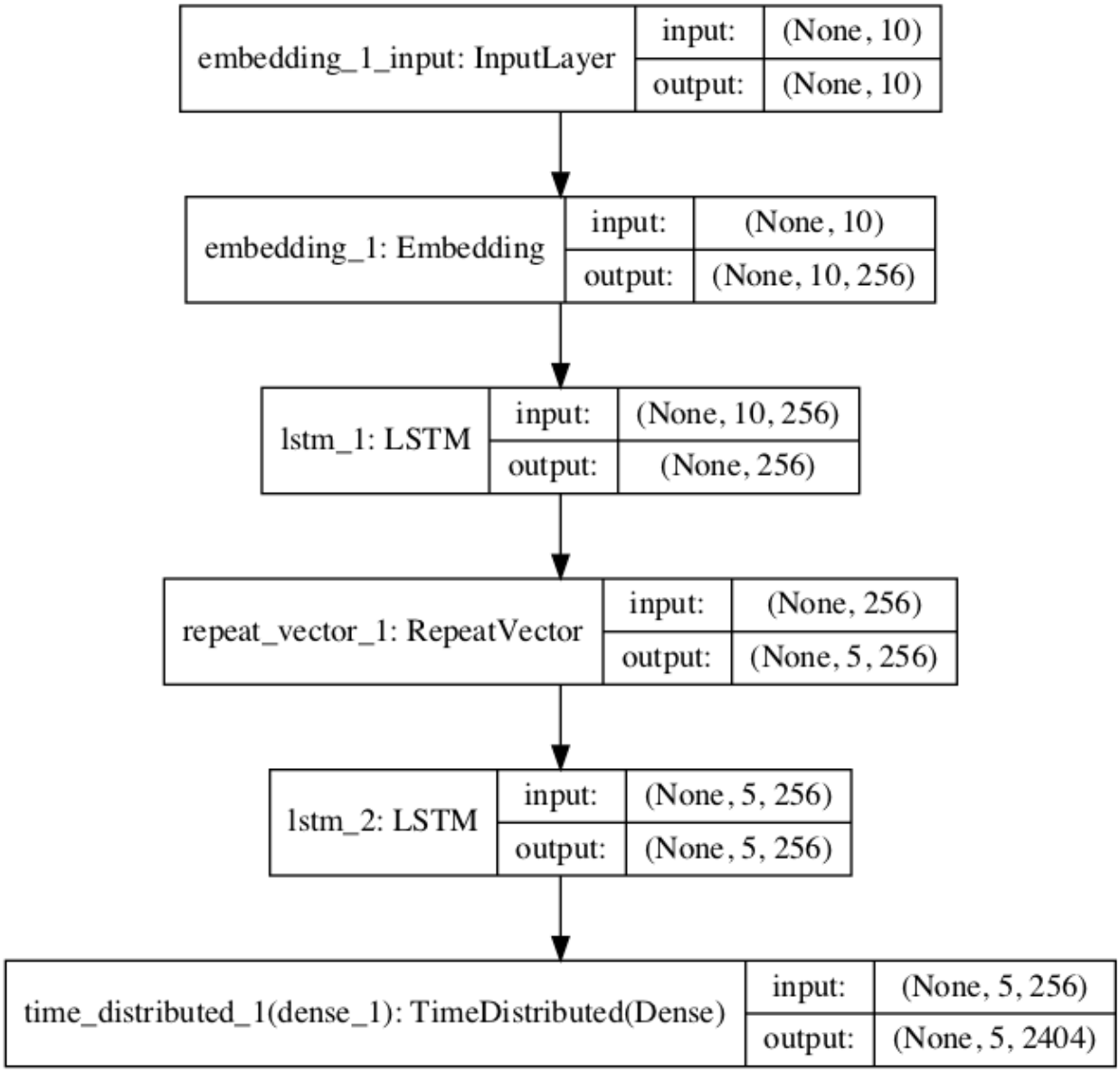
$$3686 - 2315 = 1371.$$

It quite a large number. Therefore this will be a significant inference.

Algorithms and Techniques:

A basic sequence-to-sequence model consists of two recurrent neural networks (RNNs): an *encoder* that processes the input and a *decoder* that generates the output. Sequence prediction often involves forecasting the next value in a real valued sequence or outputting a class label for an input sequence. This is often framed as a sequence of one input time step to one output time step (e.g. one-to-one) or multiple input time steps to one output time step (many-to-one) type sequence prediction problem. In this project, we will use LSTM Networks for the translation work. The Encoder-Decoder LSTM can be implemented directly in the Keras deep learning library. We can think of the model as being comprised of two key parts: the encoder and the decoder. First, the input sequence is shown to the network one encoded character at a time. We need an encoding level to learn the relationship between the steps in the input sequence and develop an internal representation of these relationships. One or more LSTM layers can be used to implement the encoder model. The output of this model is a fixed-size vector that represents the internal representation of the input sequence. The number of memory cells in this layer defines the length of this fixed-sized vector. The decoder must transform the learned internal representation of the input sequence into the correct output sequence. One or more LSTM layers can also be used to implement the decoder model. This model reads from the fixed sized output from the encoder model.

We must connect the encoder to the decoder, and they do not fit. That is, the encoder will produce a 2-dimensional matrix of outputs, where the length is defined by the number of memory cells in the layer. The decoder is an LSTM layer that expects a 3D input of [samples, time steps, features] in order to produce a decoded sequence of some different length defined by the problem. If we try to force these pieces together, we get an error indicating that the output of the decoder is 2D and 3D input to the decoder is required. We can solve this using a RepeatVector layer. This layer simply repeats the provided 2D input multiple times to create a 3D output.



In this project, we are dealing with Machine translation. The input layer is sequence of German words and we get English translated words as the output. Therefore, this concept of Long Short Term Memory Networks seems useful. The above architecture has been implemented in this project. In this architecture, the input sequence is encoded by a front-end model called the encoder then decoded word by word by a backend model called the decoder.

The model is trained using the efficient Adam approach to stochastic gradient descent and minimizes the categorical loss function because we have framed the prediction problem as multi-class classification.

Benchmark:

Machine translation has significantly evolved over time, especially in terms of accuracy levels in its output. In the present scenario, one of the well known models of this area is ***Google Translate***. It can translate any language into the desired language.
In the above stated work, we cite GNMT, ***Google's Neural Machine Translation system***. This model consists of a deep LSTM network with 8 encoder and 8 decoder layers using attention and residual connections. To improve parallelism and therefore decrease training time, its attention mechanism connects the bottom layer of the decoder to the top layer of the encoder. To accelerate the final translation speed, it employs low-precision arithmetic during inference computations. To improve handling of rare words, it divides words into a limited set of common sub-word units ("word pieces") for both input and output. This method provides a good balance between the flexibility of "character"-delimited models and the efficiency of "word"-delimited models, naturally handles translation of rare words, and ultimately improves the overall accuracy of the system.

The model in this project uses a LSTM network of single combination of encoder- decoder layers. Therefore, we would not be able to achieve the BLEU score as high as the Google Translate. However let’s analyze them anyhow.

BLEU score of Google Translate for German to English Translation: 0.86

BLEU score of our model for German to English Translation: 0.15

The model works moderately in the task of translation.

III. Methodology

Data Preprocessing:

The dataset we have requires cleaning before using in our project. As is evident from above discussion in Data Exploration, we have dataset that could result in anomalies, such as:

- There is punctuation.
- The text contains uppercase and lowercase.
- There are special characters in the German.
- There are duplicate phrases in English with different translations in German.
- The file is ordered by sentence length with very long sentences toward the end of the file.

We are now ready to clean each sentence. The specific cleaning operations we will perform are as follows:

- Remove all non-printable characters.
- Remove all punctuation characters.
- Normalize all Unicode characters to ASCII (e.g. Latin characters).
- Normalize the case to lowercase.
- Remove any remaining tokens that are not alphabetic.

We will perform these operations on each phrase for each pair in the loaded dataset. Finally, now that the data has been cleaned, we can save the list of phrase pairs to a file ready for use.

Implementation:

The project aims at translation of German words to English.
Initially we have our dataset, which is a pair of German to English translated words. We clean our data by removing punctuations, lower-upper cases, duplicates etc. We then store the cleaned data in a file. Then the data is loaded and split into training and testing sets. We tokenize the data with Keras that map words to integers, differently for English words and German words. Each input and output sequence is encoded to integers and padded to maximum phrase length. Output English sequence is one-hot encoded, as the model will predict the probability of each word in the output. The model is then trained using Adam approach to stochastic gradient descent. LSTM modeling is used. At last model is evaluated using the BLEU score. We hence get English translated words from given German words.

The project is implemented in the following steps:

- First, we load the data in a way that preserves the Unicode German characters. The function, called *load_doc(),* will load the file as a blob of text. We will perform the task of cleaning here and save our results to a file named *english-german.pkl.*

- The clean data contains a little over 150,000 phrase pairs and some of the pairs toward the end of the file are very long. We will simplify the problem by reducing the dataset to the first 10,000 examples in the file; these will be the shortest phrases in the dataset. Further, we will then stake the first 9,000 of those as examples for training and the remaining 1,000 examples to test the fit model.

- We are now ready to start developing our translation model. Let's start off by loading the datasets so that we can prepare the data. The function named *load_clean_sentences()* can be used to load the train, test, and both datasets in turn. We can use the Keras *Tokenize* class to map words to integers, as needed for modeling. We will use separate tokenizer for the English sequences and the German sequences. The function

named *create_tokenizer()* will train a tokenizer on a list of phrases. Similarly, the function named *max_length()* below will find the length of the longest sequence in a list of phrases.

- We are now ready to prepare the training dataset. Each input and output sequence must be encoded to integers and padded to the maximum phrase length. This is because we will use a word embedding for the input sequences and one hot encode the output sequences The function named *encode_sequences()* will perform these operations and return the result.

  The output sequence needs to be one-hot encoded. This is because the model will predict the probability of each word in the vocabulary as output. The function *encode_output()* will one-hot encode English output sequences.

- We are now ready to define the model. The function *define_model()* defines the model and takes a number of arguments used to configure the model, such as the size of the input and output vocabularies, the maximum length of input and output phrases, and the number of memory units used to configure the model.
- We will evaluate the model on the train and the test dataset now. We can perform mapping for each integer in the translation and return the result as a string of words. The function *predict_sequence()* performs this operation for a single encoded source phrase. We can print some of these comparisons to screen to get an idea of how the model performs in practice.

- We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed. The *evaluate_model()* function implements this, calling the above *predict_sequence()*function for each phrase in a provided dataset.

- Finally, we can train the model. We train the model for 20 epochs and a batch size of 32 examples. We use checkpointing to ensure that each time the model skill on the test set improves; the model is saved to file. We can tie all of this together and fit the neural translation model. During the run, the model will be saved to the file *model.h5*, ready for inference in the next step.

- In the last, we check the BLEU score for the model after training is completed and compare the results before and after training.

Refinement:

In this project, we are trying to maximize the BLEU score for correct predictions of English words from the given German words. The parameter I have used for refinement is the optimizer for the model. The model is trained using the efficient Adam approach to stochastic gradient descent and minimizes the categorical loss function because I have framed the prediction problem as multi-class classification.

The model performs very poorly before training process. We obtain BLEU score of about 0.02 on training set and 0.00 on testing set. This is very poor performance of the model. We can also see in the jupyter notebook that translations are very bad.

Now we train our model. After training, The BLEU score improved. The BLEU score for training dataset was 0.12 and for testing set it was 0.15.

Hence we notice that our model started to work better after training and its performance was improved, which is evident from the BLEU score.

IV. Results

Model Evaluation and Validation:

This project uses Keras as a tool for the machine translation of German to English. We build a sequence to sequence model, where German words are input and English translated words are the output. We will use an encoder-decoder LSTM model on this problem. In this architecture, the input sequence is encoded by a front-end model called the encoder then decoded word by word by a backend model called the decoder. The model is trained using the efficient Adam approach to stochastic gradient descent and minimizes the categorical loss function because we have framed the prediction problem as multi-class classification. Evaluation involves two steps: first generating a translated output sequence, and then repeating this process for many input examples and summarizing the skill of the model across multiple cases. Starting with inference, the model can predict the entire output sequence in a one-shot manner. We will also calculate the BLEU scores to get a quantitative idea of how well the model has performed.
This project works in the efficient manner and fairly translates the German words to English. We can see that some of the sentences were correctly translated, while some had issues in the translation.

Model Specifications are as follows:

Number of epochs = 20

Batch Size = 32

Optimiser = Adam

Loss Function = Categorical Cross Entropy.

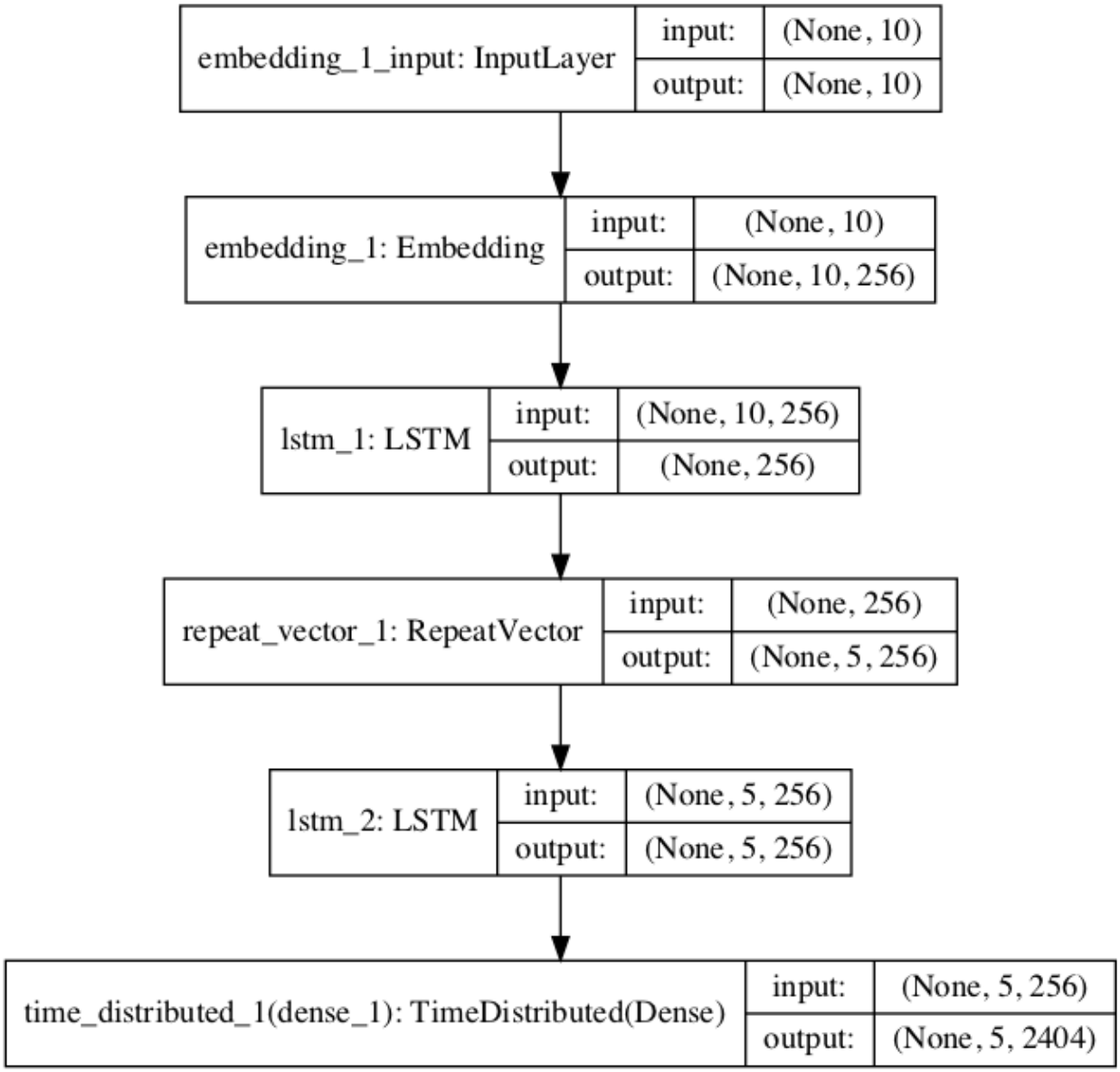
Embedding = 943616 parameters

LSTM = 525312 parameters

LSTM = 525312 parameters

Dense Layer = 594955 parameters

The parameters contributing at each layer are shown below:



Justification:

Our benchmark model was Google Translate. It uses LSTM of 8 encoders and 8 decoders. It has a BLEU score of 0.86 for the German to English Translation.

However, our model uses a LTSM network of single combination of encoders and decoders. The BLEU score we obtained was about 0.15.

It is not as good as the Google translate but performs well on the small simple translations. It works well when dealing with most common words and translation comes out to be correct. Therefore this solution holds the potential for translation of German words to English. It doesn't work excellently but it was also not very poor. It works moderately fine.

V. Conclusion

Free-Form Visualization:

```
German: das ist ein tisch,
Correct English Translation=that is a table,
Predicted English Translation=thats a a

German: sie schoss auf ihn,
Correct English Translation=she shot him,
Predicted English Translation=she adores him

German: du bist durchtrieben,
Correct English Translation=youre sharp,
Predicted English Translation=youre sharp

German: entledigen sie sich ihrer,
Correct English Translation=get rid of her,
Predicted English Translation=get rid of her

German: rate wer ich bin,
Correct English Translation=guess who i am,
Predicted English Translation=save what i am

German: tom kann antworten,
Correct English Translation=tom can answer,
Predicted English Translation=tom can yes

German: ich bin so glucklich,
Correct English Translation=im so happy,
Predicted English Translation=im am happy
```

The above image is a snap of the translation performed by our model. We can see that the model is performing translation task in an efficient manner. Earlier, we tended to obtain wrong results but now the results have improved.

Also the BLEU score has improved of the model.

We can see that das ist ein tisch was translated to thats a a, but its correct translation was thats a table. However, entledigen sie sich ihrer was correctly translated to get rid of her.
The BLEU score is about 0.15 for this model.

Reflection:

The entire project can be summarized as followed:

1. The dataset is chosen, loaded and cleaned.
2. Training and testing sets are separated.
3. We tokenize words to integers.
4. We define encoders, LSTMs, embedding layers
5. We define architecture of our model.
6. We check BLEU score before training model.
7. We train the model.
8. We inspect the performance of the model after training.
9. Our task of translating German words to English is completed.

The interesting part of this project for me is the way translation takes place. A few lines of code and a complex task of language translation is done. Woah!!!

The improvement of BLEU score was difficult for me. Tweaking of parameters took lot of patience and time to get till here.

This project fit my expectations for the problem.

Improvement:

This section lists some improvements that can be done to this project.

- **Vocabulary**. The vocabulary could be refined, perhaps removing words used less than 5 or 10 times in the dataset and replaced with "*unk*".
- **More Data**. The dataset used to fit the model could be expanded to 50,000, 100,000 phrases, or more.
- **Layers**. The encoder and/or the decoder models could be expanded with additional layers and trained for more epochs, providing more representational capacity for the model.
- **Units**. The number of memory units in the encoder and decoder could be increased, providing more representational capacity for the model.
- **Regularization**. The model could use regularization, such as weight or activation regularization, or the use of dropout on the LSTM layers.
- **Pre-Trained Word Vectors**. Pre-trained word vectors could be used in the model.
- **Recursive Model**. A recursive formulation of the model could be used where the next word in the output sequence could be conditional on the input sequence and the output sequence generated so far.

References:

- A Statistical Approach to Machine Translation, 1990.
- Review Article: Example-based Machine Translation, 1999.
- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.
- Neural Machine Translation by Jointly Learning to Align and Translate, 2014.
- Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016.
- Sequence to sequence learning with neural networks, 2014.
- Recurrent Continuous Translation Models, 2013.
- Continuous space translation models for phrase-based statistical machine translation, 2013.
- https://github.com/tensorflow/nmt
- https://www.youtube.com/watch?v=nRBnh4qbPHI&t=330s