Project Report

**NEURAL MACHINE TRANSLATION BETWEEN GERMAN AND ENGLISH**

CSE 4022 Natural Language Processing

Submitted by

15BCE0517 M.S.SANJAY

Under the guidance of

Prof. R.Sathyaraj

Bachelor of Technology

in

Computer Science and Engineering



School of Computing Science and Engineering

October 2018

**ABSTRACT:**

To build a deep neural network that converts text from English to German and vice versa and compare the performances. We are going to use Recurrent Neural network to build a neural machine translator that accepts processed text as input and outputs processed texts. These processed texts are noting but tokenized words in the sentence and they are also zero padded to make them of equal lengths. The tokenizing is different for input and output languages. The raw input and output are processed to convert it to human languages. The neural network is trained on these processed sentences.

**INTRODUCTION:**

In a machine translation task, the input already consists of a sequence of symbols in some

language, and the computer program must convert this into a sequence of symbols in another

language.

Given a sequence of text in a source language, there is no one single best translation of that text to another language. This is because of the natural ambiguity and flexibility of humanlanguage. This makes the challenge of automatic machine translation difficult, perhaps one of the most difficult in artificial intelligence: The fact is that accurate translation requires background knowledge in order to resolve ambiguity and establish the content of the sentence. Classical machine translation methods often involve rules for converting text in the source language to the target language. The rules are often developed by linguists and may operate at the lexical, syntactic, or semantic level. This focus on rules gives the name to this area of study: Rule-based Machine Translation, or RBMT. RBMT is characterized with the explicit use and manual creation of linguistically informed rules and representations. The key limitations of the classical machine translation approaches are both the expertise required to develop the rules, and the vast number of rules and exceptions required.

Neural Machine Translation-

Neural machine translation, or NMT for short, is the use of neural network models to learn a statistical model for machine translation. The key benefit to the approach is that a single system can be trained directly on source and target text, no longer requiring the pipeline of specialized systems used in statistical machine learning. Unlike the traditional phrase-based translation system which consists of many small subcomponents that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation. Neural Machine Translation by Jointly Learning to Align and Translate, 2014. As such, neural machine translation systems are said to be end-to-end systems as only one model is required for the translation.

The strength of NMT lies in its ability to learn directly, in an end-to-end fashion, the mapping from input text to associated output text.

Encoder-

The task of the encoder is to provide a representation of the input sentence. The input sentence

is a sequence of words, for which we first consult the embedding matrix. Then, as in the basic

language model described previously, we process these words with a recurrent neural network.

This results in hidden states that encode each word with its left context, i.e., all the preceding

words. To also get the right context, we also build a recurrent neural network that runs right to left, or more precisely, from the end of the sentence to the beginning. Having two recurrent

neural networks running in two directions is called a bidirectional recurrent neural network.

Decoder-

The decoder is a recurrent neural network. It takes some representation of the input context

(more on that in the next section on the attention mechanism) and the previous hidden state and

output word prediction, and generates a new hidden decoder state and a new output word

prediction.

If we use LSTMs for the encoder, then we also use LSTMs for the decoder. From the hidden

state. we now predict the output word. This prediction takes the form of a probability distribution

over the entire output vocabulary. If we have a vocabulary of, say, 50,000 words, then the

prediction is a 50,000 dimensional vector, each element corresponding to the probability

predicted for one word in the vocabulary

**LITERATURE SURVEY:**

Sequence to Sequence Learning with Recurrent Neural Networks

An overview of the model suggest it is constructed by three major components, namely the encoder, a stacked 8-layer LSTM network, the decoder network, another stacked 8-layer LSTM network, and the attention network, a one layer feedforward network connecting the two.

Neural machine translation was largely build upon Recurrent Neural Networks (RNNs), which was used as a general model in sequence to sequence learning. One important breakthrough in RNN research was the Long Short-Term Memory (LSTM) architecture proposed by Hochreiter and Schmidhuber in 1997[1]. LSTM successfully solved the problem of exploding and vanishing gradient when training conventional networks, thus capable of learning to bridge long intervals. In 2014, Sutskever et al.[2] published its results of using a 4-layer deep LSTM model with 1000 cells in each layer and 1000 dimensional word embeddings to learn English to French (Using WMT’14 Dataset) translation. This simple straightforward approach achieved a BLEU score of 34.81, surpassing past baseline of 33.30. The BLEU (bilingual evaluation understudy) score is a common metrics to measure translation quality, and is highly correlated to human judgements. The paper also reveals that LSTM learned much better if the given source sentences are reversed while the target sentence are not.

Meanwhile, Cho et al.[3] from Université de Montréal come up with a similar construction that achieved comparable results. In addition, it also proposed a much simpler recurrent unit, called Gated Recurrent Unit (GRU). This construction was later found to generate comparable results to LSTM with significantly less complexity and sometimes faster training time.

Bidirectional RNNs and Attention Mechanism

However, these simple approaches have serious limitations on the length of the sentence given for translation. More specifically, its performance would peak at sentences with length about 20 characters, and then drops as the sentence length extended.[4] Therefore, Cho et al. gave a novel approach that combined the encoding technique of Bidirectional RNNs and the Attention Mechanism in 2015. The Bidirectional RNNs obtained significant results in speech recognition in 2013, generating results slightly better than DNN and GMM-HMM baselines.[5] The technique was to read the source sentence alongside with its reverse, then concatenate them to become the input for the decoder network. With this improvement, the decoder network can benefit from a more complete description of the whole sentence.[6] As for the Attention Mechanism, it was used to enhance the encoder and the decoder’s ability to align and focus on generating its current output. Implemented by giving a weight for each bidirectional state of the sentence, and represent the input sentence as a weighted sum of these states. The weight was calculated by a feedforward network using the bidirectional state and the last output of the decoder as inputs.

This approach also solves the fact that simple representation of the sentence as a single vector is counter intuitive to the fundamentals of information theory, that is, a longer sentence should carry more information and thus have a longer encoded length. Moreover, if the encoder network was replaced by a Convolutional Neural Network (CNN), this construction can be used to generate caption for images and videos as depicted in Vinyals et al.[7] and Cho et al.[8] later in 2015.

Stacked RNN and Residual Connections

Coming to Google’s translation system GNMT,its construction was largely the same as the one Cho et al. proposed in the previous section, with modifications to its LSTM layers. Each layer’s output will be merged by its previous layer’s output to become the input for the next layer. By doing so allows the network to be expanded to more layers with feasible training speed and accuracy.

Moreover, similar to previous works, the output produced by the decoder, which was in a conditional probability format, was sent to beam search to generate the final translated sentence. However, Google’s team introduced a more sophisticated scoring based on empirical data for the beam search.

Solution to Fixed Vocabulary - Wordpiece Model

One of the most common challenges to natural language processing applications is the out of vocabulary (OOV) problem. Most language and encoding models are restricted to fixed size vocabulary, and must incorporate methods generalize to more words and lower the potential hazard of high error rate on production data. Some common solutions includes backoff methods and using a subword unit model. The same problem also applies to neural machine translation, which starts by encoding one-hot vocabulary vectors and convert it to continuous space word representations (similar to the construction of word2vec). In recent neural machine translation research, Sennrich et al. from University of Edinburgh first described a Byte Pair Encoding approach to aggregate frequent adjacent character pairs into single n-grams subwords.[9] Similar approaches have be used in speech recognition and voice search as well. A more detailed implementation was described in Schuster and Nakajima, applied in Japanese and Korean voice search.[10] The algorithm first initialize a language model with an inventory of basic subword units (i.e. characters or smallest word fragments) on the training set. Next, the create a new unit by combining two subword units and add the unit that maximizes the likelihood of the language model to the inventory, until the expected vocabulary size is matched. The Wordpiece model used in Google’s system has a subword vocabulary size of 32,000.[11]

**LITERATURE REVIEW:**

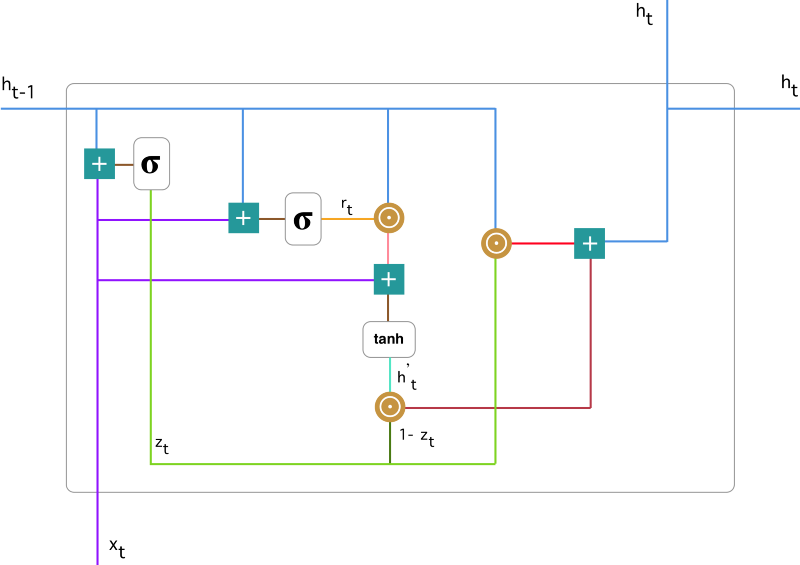
**TOOLS**:

Python,scikit library,keras library,numpy library,nltk library

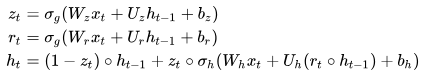
**METHODOLOGY:**

Gated Recurrent Unit:

GRUs are improved version of standard recurrent neural network. To solve the vanishing gradient problem of a standard RNN, GRU uses, so called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. They can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.



Formula:



Where,

xt is input vector

ht is output vector

zt is updated gate vector

rt is reset gate vector

W,U, and b are parameter matrices and vector

Cleaning data is the first step. Here we remove all characters that are not alphabets and space . This is so because we do not need numbers and other symbols like dollar,underscore, etc. for translation. Also we convert all characters to lower case as both upper case and lower case doesn’t make a difference for translation.

The original data contains a little over 150,000 phrase pairs and some of the pairs toward the end of the file are very long.This is a good number of examples for developing a small translation model. The complexity of the model increases with the number of examples, length of phrases, and size of the vocabulary.Although we have a good dataset for modeling translation, we will simplify the problem slightly to dramatically reduce the size of the model required, and in turn the training time required to fit the model.We simplify the problem by reducing the training dataset to 10,000 examples in the file. These will be the shortest phrases in the dataset.Further, we will then stake another 500 translated pairs as testing set.

Then we build a tokenizer to convert words to numbers. We use the Keras Tokenize class to map words to integers, as needed for modeling. We use separate tokenizer for the English sequences and the German sequences.

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 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.

We will use an encoder-decoder GRU 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.

Summary of model for german to English :

Layer (type) Output Shape Param #

=================================================================

embedding\_1 (Embedding) (None, 10, 100) 389700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_1 (GRU) (None, 150) 112950

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 150) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 100) 15100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

repeat\_vector\_1 (RepeatVecto (None, 5, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_2 (GRU) (None, 5, 100) 60300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

time\_distributed\_1 (TimeDist (None, 5, 2733) 276033

=================================================================

Total params: 854,083

Trainable params: 854,083

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Summary of model from English to German:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_2 (Embedding) (None, 5, 100) 273300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_3 (GRU) (None, 150) 112950

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 150) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 100) 15100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

repeat\_vector\_2 (RepeatVecto (None, 10, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_4 (GRU) (None, 10, 100) 60300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

time\_distributed\_2 (TimeDist (None, 10, 3897) 393597

=================================================================

Total params: 855,247

Trainable params: 855,247

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

**EXPERIMENT AND RESULT:**

We evaluate the model on the test dataset.

The model should perform very well on the train dataset and ideally have been generalized to perform well on the test dataset.

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.

This will be a sequence of integers that we can enumerate and lookup in the tokenizer to map back to words.

We perform this mapping for each integer in the translation and return the result as a string of words.

Next, we repeat this for each source phrase in a dataset and compare the predicted result to the expected target phrase in English.

We also calculate the NIST scores to get a quantitative idea of how well the model has performed.

The NIST metric is derived from the BLEU evaluation criterion but differs in one fundamental aspect: instead of n-gram precision the information gain from each n-gram is taken into account. The idea behind this is to give more credit if a system gets an n-gram match that is difficult, but to give less credit for an n-gram match which is easy

German to English NIST score: 0.00014994352556707308

English to German NIST score: 1.8521987516760904e-05

**CONCLUSION:**

The possible reason German to English translation is better because there are gender for objects in German, unlike English that doesn’t show any gender bias. Thus the articles, negation words and noun words in German have a corresponding male and female and plural forms also. So we can expect the model to confuse between them. This problem can be alleviated if a very large number of training set is used.

The possible extensions to our project are:

Data Cleaning. Different data cleaning operations could be performed on the data, such as not removing punctuation or normalizing case, or perhaps removing duplicate English phrases.

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.

Input Order. The order of input phrases could be reversed, which has been reported to lift skill, or a Bidirectional input layer could be used.

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.

In just a few years, Neural Machine Translation (NMT)(Bahdanau et al., 2015; Cho et al., 2014) has become the main approach to Machine Translation as well as one of the most successful application of Deep Learning to NLP. It leverages powerful machine learning techniques to train complex translation models in an end-to-end manner. Although this area of research is pretty new, the many recent developments combined with the practical difficulties of deep learning can make it difficult for a researcher lacking the background and practical experience to develop state-of-the-art models. The recent advancements in the field of artificial intelligence gives us a good chance to conduct Neural machine translation research and proceed to further advancements in NMT.

**REFERENCES**:

[1]. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation, 1997.

[2]. I. Sutskever, O. Vinyals and Q. Le. Sequence to Sequence Learning with Neural Networks. NIPS, 2014. 3. K. Cho, D. Bahdanau, F. Bougares, H, Schwenk and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP, 2014.

[3]. K. Cho, B. Merrienboer, D. Bahdanau and Y. Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST-8.

[4]. A. Graves, N. Jaitly and A. Mohamed. Hybrid Speech Recognition with Deep Bidirectional LSTM. ASRU, 2013.

[5]. D. Bahdanau, K. Cho and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR, 2015.

[6]. O. Vinyals, A. Toshev, S. Bengio and D. Erhan. Show and Tell: A Neural Image Caption Generator. arXiv:1411.4555v2 [cs.CV] 20 Apr 2015.

[7]. K. Cho, A. Courville and Y. Bengio. Describing Multimedia Content using Attention-based Encoder--Decoder Networks. IEEE Transactions on Multimedia, 2015.

[8]. R.Sennrich, B. Haddow, A. Birch. Neural Machine Translation of Rare Words with Subword Units. ACL, 2016.

[9]. M. Schuster and K. Nakajima. Japanese and Korean voice search. ICASSP, 2012.

[10]. Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi and the Google Brain team. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144v2 [cs.CL] 8 Oct 2016