VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE 4022 - Natural Language Processing

Encoder-Decoder Based Machine Translation using Attention Model

(SLOT - A2)

Team ID - 4

19BCE0688 - Debangan Mandal

19BCE0747 - Shobhit Gupta

19BCE0801 - Aryan Sridhar

19BCE0735 - Anand Agarwal

Under the guidance of

Prof. Rajeshkannan R.

1. Abstract

The aim of our project is to automate the language translation problem to overcome the language barrier among countries and also states within the country. Our model will perform the various features translations required for achieving our aim. The model recognizes text in one language to another user defined language to communicate in an expressive manner. The language translation is efficient and helps in breaking down the language down to its corpus and re-arranging for the next stage. The architecture we will be using, it will help us understand the in depth knowledge of the application of recurrent neural networks. These methods are actually used in Google's language translators, obviously with much more precision and better architecture. According to US Foreign services, Japanese language is the hardest to interpret for native English Speakers. So we have decided to cater to the problem by designing an Encoder Decoder Based Model to translate Japanese to English Language.

2. Introduction

The idea of Machine translation comes from the presence of linguistic diversity amongst the people. In most countries, all the states generally have the same language. According to US Foreign services, Japanese language is the hardest to interpret for native English Speakers. This intense linguistic division can sometimes be a hurdle for communication. It is not possible for human beings to keep track and learn each and every language. There has been an ample amount of research done, where many scientists have used many different ways to achieve their target of efficient machine translation. Machine Translation can be described as part of computational linguistics that studies the use of software tools to translate text or speech from one language to another.

Human and machine translation each have their share of challenges. For example, no two individual translators can produce identical translations of the same text in the same language pair, and it may take several rounds of revisions to meet customer satisfaction. But the greater challenge lies in how machine translation can produce publishable quality translations.

There are many ways like, Rule Based Machine Translation (RBMT), Corpus Based MT and Hybrid MT. Rule Based Machine Translation takes into account the linguistic rules based on morphological grammars keeping in mind the linguistic rules of the target and source languages. This generates target language sentences on the basis of syntactic, semantic and morphological rules and regularities of each language taken under consideration.

Corpus Based Machine Translation requires a huge amount of bilingual data that contains the source language and the translated language as well. Corpus based on which the models would be learning are POS tagged and parsed.

There are few types of Corpus Based Machine Translation:

1. Statistical Based Machine Translation :- It is basically referred to as the use of statistical models that learn to text from source language to the target language using a huge amount of corpus language data.

2. Example Based Machine Translation :- Also known as Memory based translation, in which a set of sentences from source languages are given and the sentences used in the target language having the same meaning are also present to use. Thus this point to point mapping is possible, and the dataset is feeded into the model, which can be a basic machine learning model or neural network model.

Neural Network Model is the most recently used and the most booming application for the problem of Machine Translation. The problem and with the presence of the correct dataset, we would be able to get good results with the help of Neural Network.

3. Literature Survey

S.No	Name	Year	Methodology	Observation	Result
1.	Attention-based Multimodal Neural Machine Translation	2016	Encoder - decoder framework with attention model , LSTM	With attention, the decoder can **refresh** it's memory to focus on source words that may help to translate the correct words rather than only seeing the last state of the sentences	So as an output this attention model will greatly benefit in improving our BLEU score for the model
2.	Neural Machine Translation by Jointly Learning to Align and Translate	2015	Encoder - decoder neural network	Neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences	By introducing an extension to the encoder–decoder model which learns to align and translate jointly
3.	Contrastive Attention Mechanism for Abstractive Sentence Summarization	2019	Contrastive Attention Mechanism	The seq2seq model is normally composed of an encoder-decoder architecture, where the encoder processes the input sequence and encodes/compresses the information into a context vector (or "thought vector") of fixed length.	Using Transformer as a strong baseline, experiments on three benchmark data sets show that the proposed contrastive attention mechanism significantly

					improves the performance, advancing the state-of-the-art performance for the task
4.	Abstractive Summarization Using Attentive Neural Techniques	2018	Attention mechanism	In deep learning models, attention allows a decoder to focus on different segments of an input while stepping through output regions. The resulting text must be much more condensed than the original.	All generated summaries are constrained to a fixed maximum length so that tested models must learn how to decide what information should be reproduced.
5.	Sequence to Sequence Learning with Neural Networks	2014	LSTM based approach on machine translation	DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. Domain-independent method that learns to map sequences to sequences would be useful.	A large deep LSTM with a limited vocabulary can outperform a stan- dard SMT-based system whose vocabulary is unlimited on a large-scale MT task.
6.	Long Short-Term Memory Recurrent Neural Network Architectures	2014	Two layer LSTM Recurrent Neural Networks With Seq2Seq	A critical disadvantage of this fixed-length context vector design is the inability of the system to retain longer sequences. The attention mechanism	Deep LSTM RNN architectures achieve state of-the-art performance for large scale

	for Large Scale Acoustic Modeling		model	was created to resolve this problem of long dependencies.	acoustic modeling
7.	Attention in Natural Language Processing	2021	RNN search uses attention for machine translation	The attention mechanism can also be used as a tool for interpreting the behavior of neural architectures, which are notoriously difficult to understand	Attention can be applied to different input parts, different representations of the same data, or different features, to obtain a compact representation of the data as well as to highlight the relevant information.
8.	Selective Encoding for Abstractive Sentence Summarizatio n	2017	Selective Encoding for Abstractive Sentence Summarizatio n (SEASS)	They were surprised by the ability of the LSTM to correctly translate very long sentences. We were initially convinced that the LSTM would fail on long sentences due to its limited memory.	The proposed selective mechanism, we build an end-to-end neural network summarization model which consists of three phases: encoding, selection, and decoding
9.	Fundamentals of Recurrent Neural Network	2020	RNN and LSTM network	The LSTM network is a type of an RNN, and since the RNN is a simpler system, the	The main contribution up to this point has been our unique

	(RNN) and Long Short-Term Memory (LSTM)			intuition gained by analyzing the RNN applies to the LSTM network as well.	pedagogical approach for analyzing the RNN and Vanilla LSTM systems from the Signal Processing perspective
10.	A Critical Review of Recurrent Neural Networks for Sequence Learning	2015	LSTM and Bidirectional RNN's	Neural networks have limitations most notably, they rely on the assumption of independence among the training and test examples. After each example (data point) is processed, the entire state of the network is lost. If each example is generated independently, this presents no problem.	LSTMs and BRNNs have set records in accuracy on many tasks in recent years, it is noteworthy that advances come from novel architectures rather than from fundamentally novel algorithms.

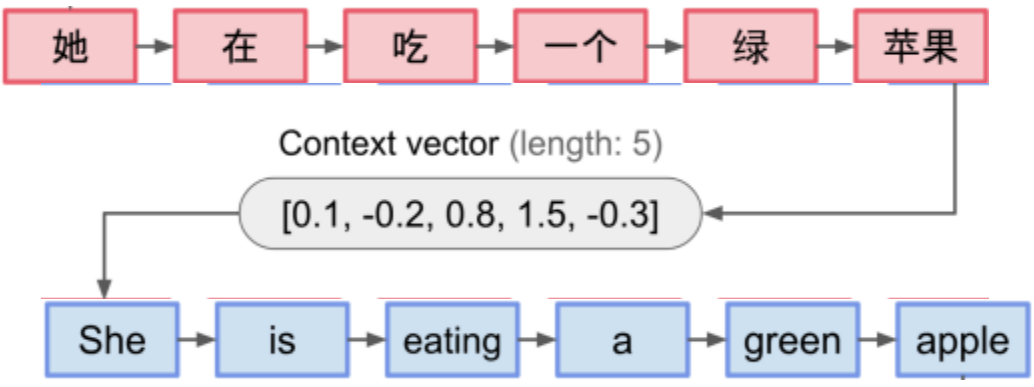
4. Problem Description

When we go around the world we tend to find some very interesting personalities, be it italian, or german and even japanese. We all know english is the very basic language that is used the most for communicating around the world. That's why it is also known as the shopkeeper's language. But sometimes we still have some communication difficulties, in various countries. This is where we would like to introduce our model. We have chosen two languages, japanese and english, and now the model will help us translate the japanese language into english sentences. So from a high level, our model uses Encoder-Decoder Models, and then for more accuracy and interpreting longer sentences, we are using Attention models.

So we have to achieve the best possible method for translating from Japanese to English. For Example:

あなたは学校に行くつもりか? → are you going to school?

A pictorial description of our project :



4.1. Architecture Diagram

The architecture we will be using is called the Encoder Decoder Model using Attention Modeling.

The Encoder Decoder Model is an architecture for recurrent neural networks that is proving to be powerful on a host of seq2seq (sequence to sequence) prediction problems in the field of natural language processing.

From the diagram we can see, there are few boxes that are drawn in the diagram. Each of them are the LSTM units that are the building blocks of the RNN architecture used. The LSTMs are connected sequentially, and have been changed in some way or the other to have our desired outcome.

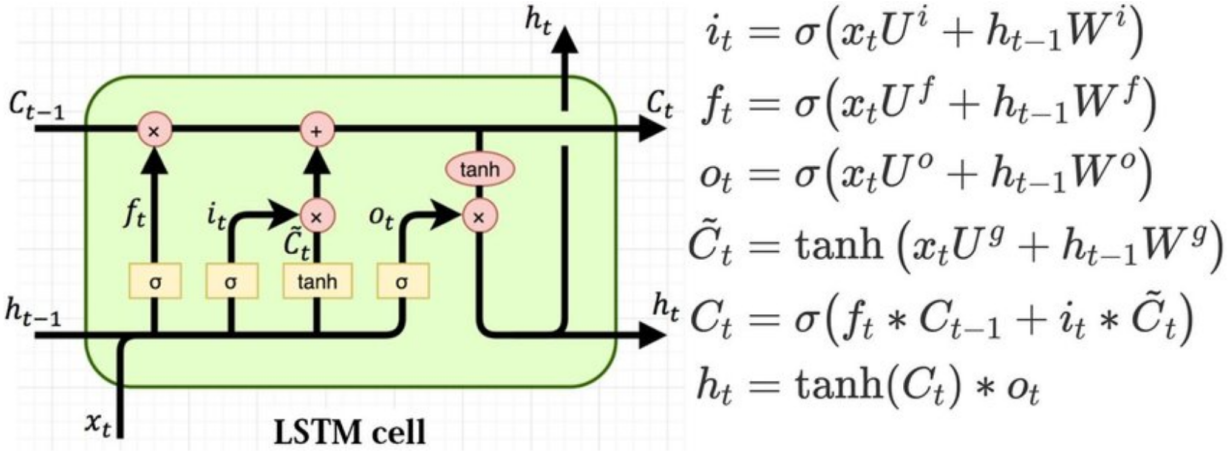


Fig 1: LSTM Cell

We can deduce that Neural Machine Translation is a Seq2Seq problem. Specifically it is a many-to-many type problem, with a sequence of several elements both at the input and at the output, and the encoder-decoder architecture for recurrent neural networks is the standard method.

This model consists of two subnetworks, the encoder and the decoder. The encoder, on the left hand side of the diagram, receives the sequences from the source language as inputs and produces, as a result, a compact representation of the input sequence, trying to summarize or condense all of its information. Then that output becomes an input or initial state to the decoder, which can also receive another external input.

At each time step, the decoder generates an element of its output sequence based on the input received and its current state, as well as updating its own state for the next time step.

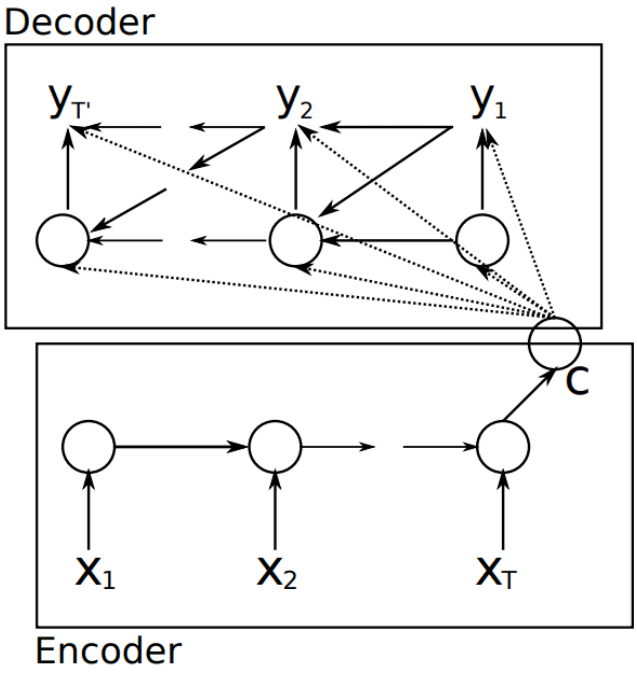


Fig 2 : Encoder - Decoder Architecture

The problem in using Encoder decoder architecture:

The critical point of this model is how to get the encoder to provide the most complete and meaningful representation of its input sequence in a single output element to the decoder because this vector or state is the only information the decoder will receive from the input to generate the corresponding output. The longer the input, the harder it'll be to compress into a single vector.

This will be solved by the ***Attention Mechanism.***

The previously described model based on RNNs has a serious problem when working with long sequences because the information of the first tokens is lost or diluted as more tokens are processed. The context vector has been given the responsibility of encoding all of the information in a given source sentence into a vector of a few hundred elements. It made it challenging for the models to deal with long sentences.

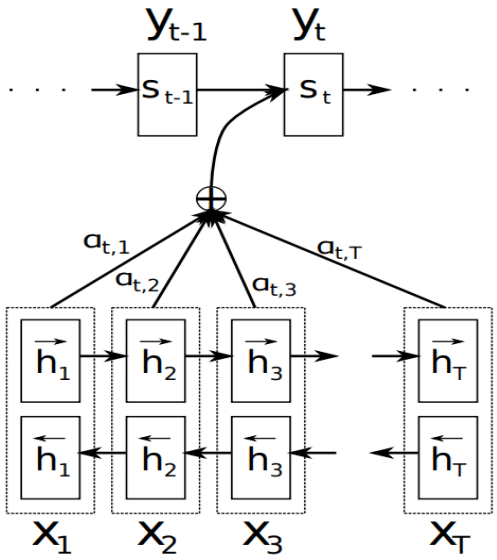


Fig 3 : Attention Model

A research paper introduced the solution:

Neural Machine Translation by Jointly Learning to Align and Translate

So the paper mentioned a technique called *attention,* which highly improved the quality of machine-translation systems. *"Attention allows the model to focus on the relevant parts of the input sequence as needed, accessing all the past hidden states of the encoder, instead of just the last one".* At each decoding step, the decoder gets to look at any particular state of the encoder and can selectively pick out specific elements from that sequence to produce the output.

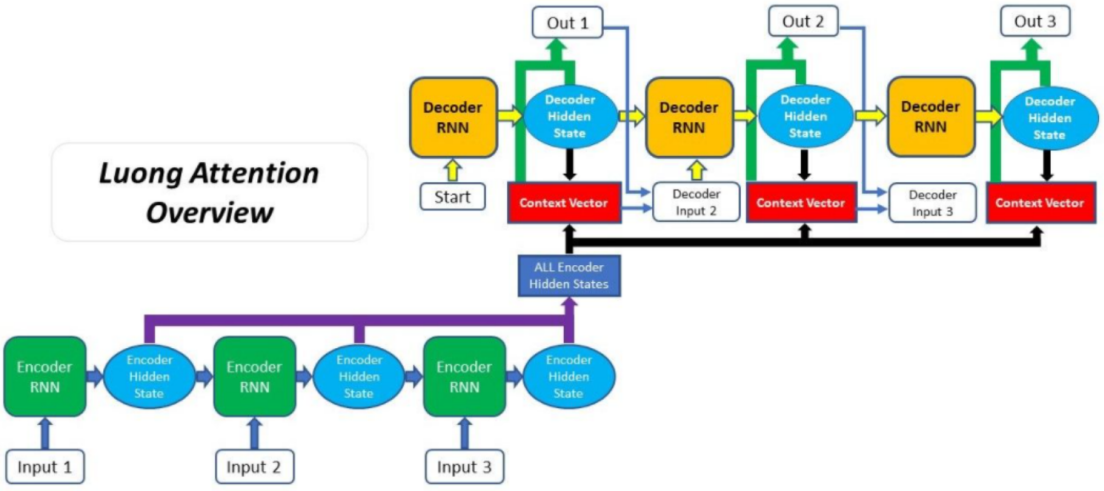
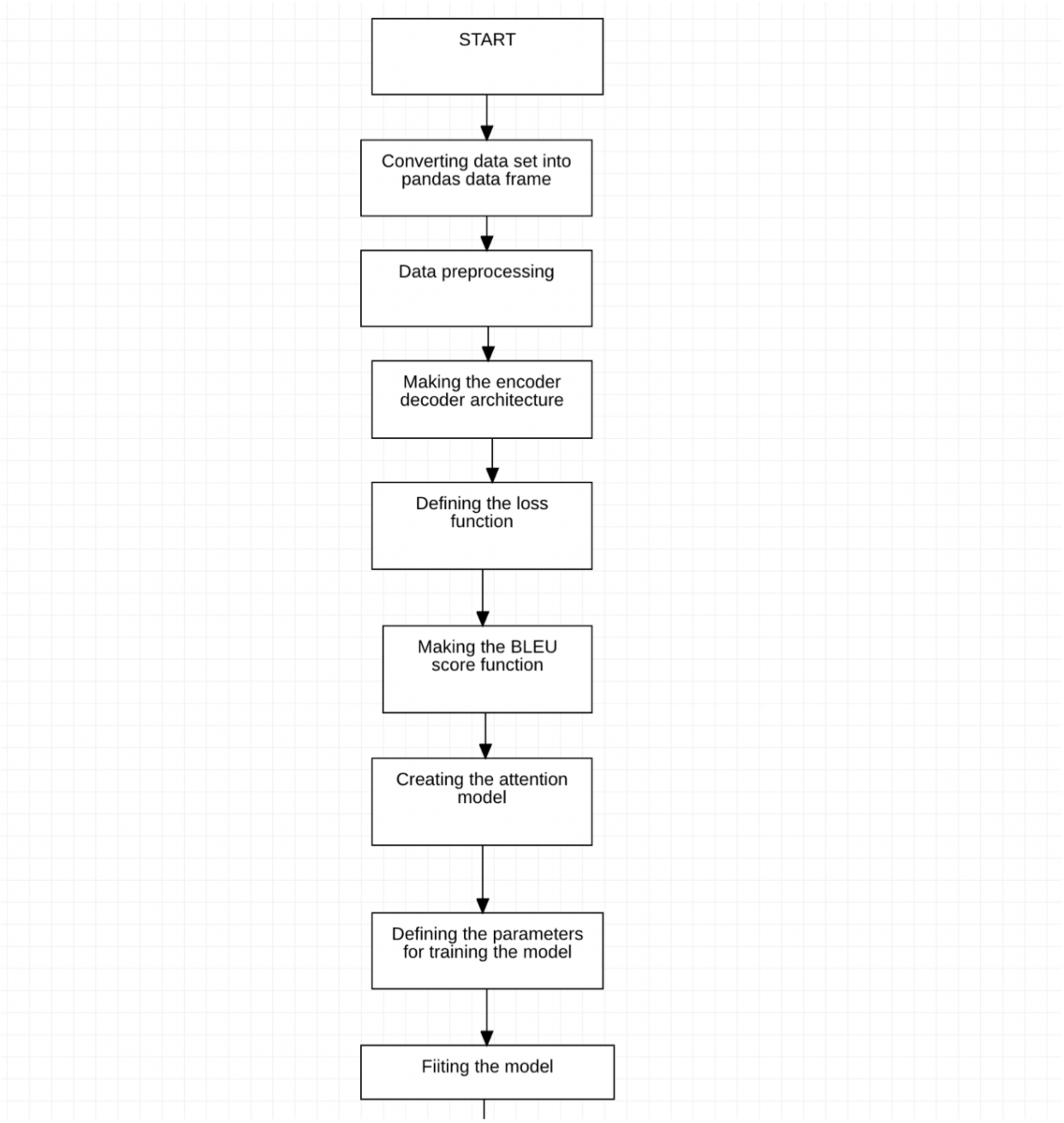


Fig 4 : Luong Attention Overview

4.2. Flow Diagram



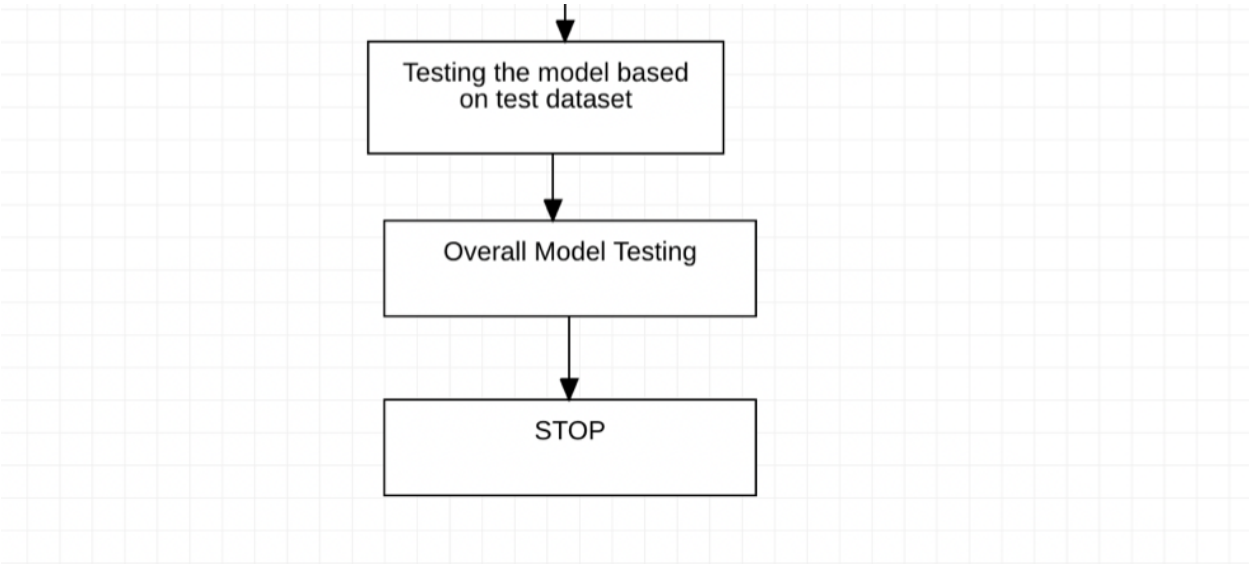


Fig 6 : Project Flow Diagram

4.3. Pseudocode

string<= input_text

data <= dataset

for all reads read do

for token in read.tokenize do if token in data then

    token idx = getindex(data, token)

    bg feature[token idx]++ else

continue

end if

    end for tokenize(string)

encoder<=embedding_dims, encoder_units

Building the encoder

Deinitializing the states

decoder <= embedding_dim, decoder_units

Building the decoder

DeInitializing the states

Initialize the optimizer function (adam optimizer)

BLEU Score function

Decoder_inputs and encoder_outputs on the line, calculate the bleu score

Using the bleu() function

Create the attention model

Fit the attention model and re-evaluate the weights using the BLEU Score

Testing the BLEU score for Epochs

Fitting the model onto the training model

Testing the model

5. Experiments and Results

5.1 Dataset Description

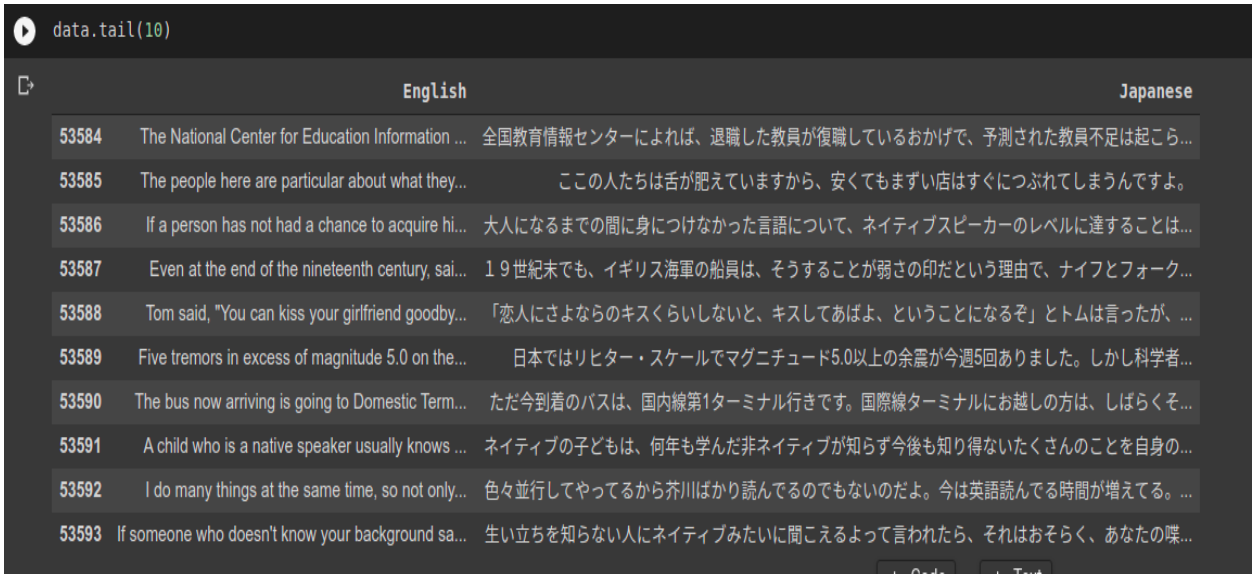
Natural Language Processing is a very hot topic in the Artificial Intelligence field. We are using the Japanese-English Bilingual Corpus, which aims mainly at supporting research and development relevant to high-performance multilingual machine translation, information extraction, and other language processing technologies.

It is a precise and large-scale corpus containing about 50,000 pairs of manually translated sentences. The three-step translation process has been clearly recorded. Enables observation of how translations have been elaborated so it can be applied for uses such as research and development relevant to translation aids and error analysis of human translation.

Sample Dataset:

```
[ ] data.head(10)
```

	English	Japanese
0	Go.	行け。
1	Go.	行きなさい。
2	Hi.	こんにちは。
3	Hi.	もしもし。
4	Hi.	やっほー。
5	Hi.	こんにちは！
6	Run.	走れ。
7	Run.	走って！
8	Who?	誰？
9	Wow!	すごい！



```
data.tail(10)
```

	English	Japanese
53584	The National Center for Education Information ...	全国教育情報センターによれば、退職した教員が復職しているおかげで、予測された教員不足は起こら...
53585	The people here are particular about what they...	ここの人たちは舌が肥えていますから、安くてもまずい店はすぐにつぶれてしまうんですよ。
53586	If a person has not had a chance to acquire hi...	大人になるまでの間に身につけなかった言語について、ネイティブスピーカーのレベルに達することは...
53587	Even at the end of the nineteenth century, sai...	１９世紀末でも、イギリス海軍の船員は、そうすることが弱さの印だという理由で、ナイフとフォーク...
53588	Tom said, "You can kiss your girlfriend goodby...	「恋人にさよならのキスくらいしないと、キスしてあばよ、ということになるぞ」とトムは言ったが、...
53589	Five tremors in excess of magnitude 5.0 on the...	日本ではリヒター・スケールでマグニチュード5.0以上の余震が今週5回ありました。しかし科学者...
53590	The bus now arriving is going to Domestic Term...	ただ今到着のバスは、国内線第1ターミナル行きです。国際線ターミナルにお越しの方は、しばらくそ...
53591	A child who is a native speaker usually knows ...	ネイティブの子どもは、何年も学んだ非ネイティブが知らず今後も知り得ないたくさんのことを自身の...
53592	I do many things at the same time, so not only...	色々並行してやってるから芥川ばかり読んでるのでもないのだよ。今は英語読んでる時間が増えてる。...
53593	If someone who doesn't know your background sa...	生い立ちを知らない人にネイティブみたいに聞こえるよって言われたら、それはおそらく、あなたの喋...

Fig 7 : Dataset

Link for dataset: http://www.manythings.org/anki/

How the Data Looks in the Data Set:

English + TAB + The Other Language + TAB + Attribution

```
This work isn't easy.	この仕事は簡単じゃない。	CC-BY 2.0 (France) Att
Those are sunflowers.	それはひまわりです。	CC-BY 2.0 (France) Att
Tom bought a new car.	トムは新車を買った。	CC-BY 2.0 (France) Att
This watch is broken.	この時計は壊れている。	CC-BY 2.0 (France) Att
```

Fig 8 : Dataset Format

From the above dataset we have applied various preprocessing methods like tokenization and a particular built-in function called the janome tokenizer.

5.1.1. Methodology

Data Preprocessing

The text sentences are almost clean, they are plain texts, so we only need to remove accents, lowercase the sentences, and replace everything with a space except (a-z, A-Z,.,?,! and ,).

There are some misspelled words that are used in everyday life. We need to de-abbreviate or deconcate them as well. After deconcating, we need to add the Start of Sentence and End of Sentence Token to each of the statements for both source and target language.

With a slight analysis we can understand that Japanese sentences tend to have longer word length than English sentences.

Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.

Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

For Example:

Consider the string: This is a cat

Expected Output: [‘This’, ‘is’, ‘a’, ‘cat’]

There are numerous uses of doing this. We can use this tokenized form to:

- Count the number of words in text
- Count number of times a particular word is present

Now we Tokenize the data to convert the raw text into a sequence of integers. First we create a Tokenizer object from the Keras library and fit it to the english texts.

Whereas for the Japanese texts, we have the **Janome**. Janome is a Japanese morphological analysis engine purely built in Python, which includes a built-in dictionary and the language model.

Then we calculate the size of the english and japanese vocabularies in the given dataset. Then we also calculate the maximum length of English and Japanese sentences in the given data, which will help us add the padding in the sentences.

Padding the sentences: We need to pad zeros at the end of the sequences so all sequences have the same length. Otherwise, we won't be able to train the model in batches.

Model Description

First we need to know what kind of model we will be using. As described earlier, we are implementing our solution using the Encoder decoder model. So in the encoder model (i.e, the encoder function) we are initializing the embedding dimensions, the input shapes, and also the lstm using **tf.compact.v1.keras.layers.CuDNNLSTM** function.

After that we need to disable the output of the Encoder part of the model. The output of an LSTM cell or layer of cells is called the hidden state.

This is perplexing because each LSTM cell has an internal state termed the cell state, or c, that is not output.

In most cases, we don't need to access the cell state unless we're working on complex models that require succeeding layers' cell states to be initialised with the final cell state of a previous layer, such as in an encoder-decoder model.

Keras gives the LSTM layer the return state argument, which gives it access to the hidden state output (state h) and the cell state (state c).

For Example:

```
self.lstm_output, self.lstm_state_h, self.state_c =
self.lstm(input_embedd, initial_state=[state_h, state_c])
```

This may look confusing because both lstm1 and *state_h* refer to the same hidden state output.

We have disabled the hidden state (state_h) by putting the tensor values to zeros.

Code written :

```
   def initialize_hidden_state(self, batch_sz):

       return [tf.zeros((batch_sz, self.enc_units)), tf.zeros((batch_sz,
self.enc_units))]
```

The cell state will be proceeded further and thus not changed or hampered with, based on the fact that it contains the tensors that are the memory states of the processed inputs. These tensors will be further decoded by the decoder to translate into the Japanese sentences.

Use LSTM for both the encoder and the decoder if you used it for the encoder. However, it is a little more complicated than the encoder network. The decoder is said to be in a "awake state." It is aware of the words you have generated thus far, as well as the prior hidden state. The first layer of the decoder is initialised by generating the output using the context vector 'C' from the encoder network. Then, at the start, a special token is applied to identify the output generation. At the end, it uses a similar token. The stacked LSTM layers are used to generate the first output word.

The decoder_target_data is like decoder_input_data, the only difference is that the decoder_target_data is offset by one timestamp. The decoder_target_data[:, t, :] is the same as decoder_input_data[:, t + 1, :] .

The term "decode" refers to the process of converting a coded message into understandable English. The drawing will be converted into a word by the second person in the Pictionary team. The decoder's job in the machine learning model is to turn the two-dimensional vector into the output sequence, which is an English sentence. To predict the English term, it is also developed with RNN layers and a thick layer.

Transformers and attention have been successfully used in a variety of tasks in NLP, including text comprehension, abstractive summarization, and word completion, among others.

The attention mechanism arose organically as a result of challenges involving time-varying data (sequences). So, given we're working with "sequences," let's start with a machine-learning-based solution. In the overall problem of dealing with sequences, attention became popular.

There are some very obvious limitations of the basic Encoder-Decoder Model

The intermediate representation is unable to encode all of the input timesteps. The bottleneck problem is what it's called. All of the information about the source sentence must be captured by the vector z.

This is theoretically feasible, according to mathematics. In practise, however, the amount of time we can see in the past (the so-called reference window) is limited. RNNs have a tendency to lose knowledge from previous timesteps.

Attention was born in order to address two things on the Seq2seq model:

- we need to form a **direct connection** with each timestamp.
- the stacked RNN layer usually create the well-know **vanishing gradient problem**

This concept was first introduced in the context of computer vision. We can learn to amass information about a shape and identify the image by glancing at different areas of the image (glimpses), according to Larochelle and Hinton.

Later, the same idea was used in sequences. We can look at all of the words at once and learn to "pay attention" to the ones that are correct for the task at hand.

And there you have it. This is what we now refer to as attention, which is just a concept of memory gained by paying attention to a variety of inputs throughout time.

In the encoder-decoder RNN case, given previous state in the decoder as

yi-1 and the the hidden state h_1,h_2, h_{n} we have something like this:

ei = attention_net(y_i-1, h)

The index i indicates the prediction step. Essentially, we define a score between the hidden state of the decoder and all the hidden states of the encoder.

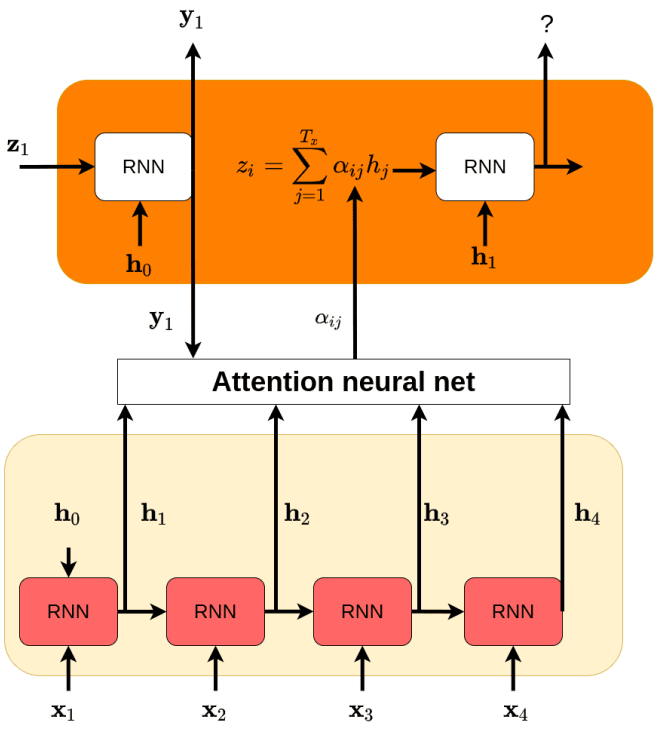


Fig 9 : RNN

In both the equation and the diagram, we utilised the symbol alpha! Why?

Because we want it to have several additional properties: a) it should be a probability distribution, and b) the scores should be far apart. The latter, which is nothing more than our well-known softmax, results in more confidence predictions.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Finally, here is where the new magic will happen:

$$z_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j$$

Attention is defined as the weighted average of values in theory. The weighting, however, is a learned function this time! We can conceive about alpha i,j as data-dependent dynamic weights intuitively. As a result, it is self-evident that we require a concept of memory, and as previously said, attention weight stores the memory accumulated via time.

BLEU Score:

We need a single real number score to feed into our loss function if we wish to use machine learning to develop a machine translation system. We can determine the distance between the two if we also know the potential best score. This allows us to provide feedback to our system while it is training, such as if a prospective adjustment will enhance a translation by bringing the score closer to the ideal score, and to compare trained systems using the same task. Generally, BLEU scores are based on an average of unigram, bigram, trigram and 4-gram precision

BLEU, on the other hand, has several advantages. The ones that matter the most to those who work in NLP have to do with how convenient it is for researchers.

- It’s fast and easy to calculate, especially compared to having human translators rate model output.
- It's ubiquitous. It helps in comparing the various benchmarks on the same task

5.2. Sample Outputs

5.2.1 Sample Training Variables, Scores and Epochs

```
[ ]  BUFFER_SIZE = len(eng_tr)
     BATCH_SIZE = 64
     N_BATCH = BUFFER_SIZE//BATCH_SIZE
     embedding_dim = 128
     units = 256
     EPOCHS = 10
     print(N_BATCH)

     dataset = tf.data.Dataset.from_tensor_slices((jpn_tr, eng_tr)).shuffle(BUFFER_SIZE)
     dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)

     812
```

```
[ ] tf.keras.backend.clear_session()
    model = MyModel(jpn_vocab, eng_vocab, embedding_dim, units)
    model.fit(dataset, BATCH_SIZE, EPOCHS)
    model.save_weights('jpn_to_eng.h5')
```

```
Epoch 1 Batch 0 Loss 1.3290
Epoch 1 Batch 100 Loss 1.0129
Epoch 1 Batch 200 Loss 0.8957
Epoch 1 Batch 300 Loss 0.9438
Epoch 1 Batch 400 Loss 0.8744
Epoch 1 Batch 500 Loss 0.8272
Epoch 1 Batch 600 Loss 0.8737
Epoch 1 Batch 700 Loss 0.7378
Epoch 1 Batch 800 Loss 0.7313
Epoch 1 Loss 0.8929 BLEU-score 0.19660661815241223
Time taken for 1 epoch 378.5246226787567 sec

training completed...!
returning gradients...
Epoch 2 Batch 0 Loss 0.7552
Epoch 2 Batch 100 Loss 0.7504
Epoch 2 Batch 200 Loss 0.8018
Epoch 2 Batch 300 Loss 0.7269
Epoch 2 Batch 400 Loss 0.7559
Epoch 2 Batch 500 Loss 0.6856
Epoch 2 Batch 600 Loss 0.6749
Epoch 2 Batch 700 Loss 0.6682
Epoch 2 Batch 800 Loss 0.6886
Epoch 2 Loss 0.7323 BLEU-score 0.3543531002670155
Time taken for 1 epoch 323.60630893707275 sec

training completed...!
returning gradients...
Epoch 3 Batch 0 Loss 0.6334
Epoch 3 Batch 100 Loss 0.6986
Epoch 3 Batch 200 Loss 0.6413
Epoch 3 Batch 300 Loss 0.6773
Epoch 3 Batch 400 Loss 0.5940
Epoch 3 Batch 500 Loss 0.6322
Epoch 3 Batch 600 Loss 0.6122
Epoch 3 Batch 700 Loss 0.6454
Epoch 3 Batch 800 Loss 0.6046
Epoch 3 Loss 0.6369 BLEU-score 0.38669539555546967
Time taken for 1 epoch 323.12049412727356 sec
```

```
Epoch 8 Batch 0 Loss 0.4063
Epoch 8 Batch 100 Loss 0.4130
Epoch 8 Batch 200 Loss 0.3717
Epoch 8 Batch 300 Loss 0.3848
Epoch 8 Batch 400 Loss 0.4015
Epoch 8 Batch 500 Loss 0.3877
Epoch 8 Batch 600 Loss 0.3779
Epoch 8 Batch 700 Loss 0.3875
Epoch 8 Batch 800 Loss 0.3491
Epoch 8 Loss 0.3964 BLEU-score 0.44449992042233283
Time taken for 1 epoch 324.0923023223877 sec

training completed...!
returning gradients...
Epoch 9 Batch 0 Loss 0.3445
Epoch 9 Batch 100 Loss 0.3644
Epoch 9 Batch 200 Loss 0.3307
Epoch 9 Batch 300 Loss 0.3180
Epoch 9 Batch 400 Loss 0.3894
Epoch 9 Batch 500 Loss 0.3245
Epoch 9 Batch 600 Loss 0.3653
Epoch 9 Batch 700 Loss 0.3909
Epoch 9 Batch 800 Loss 0.3521
Epoch 9 Loss 0.3636 BLEU-score 0.455224320380367
Time taken for 1 epoch 323.47976183891296 sec

training completed...!
returning gradients...
Epoch 10 Batch 0 Loss 0.3582
Epoch 10 Batch 100 Loss 0.3339
Epoch 10 Batch 200 Loss 0.3286
Epoch 10 Batch 300 Loss 0.2964
Epoch 10 Batch 400 Loss 0.4019
Epoch 10 Batch 500 Loss 0.4105
Epoch 10 Batch 600 Loss 0.3119
Epoch 10 Batch 700 Loss 0.3398
Epoch 10 Batch 800 Loss 0.3607
Epoch 10 Loss 0.3335 BLEU-score 0.45252387741163547
Time taken for 1 epoch 323.6990990638733 sec

training completed...!
returning gradients...
```

As you can see the BLEU score has increased as more epochs are executed for the code.

Interpretation

Trying to compare BLEU scores across different corpora and languages is strongly discouraged. Even comparing BLEU scores for the same corpus but with different numbers of reference translations can be highly misleading.

However, as a rough guideline, the following interpretation of BLEU scores (expressed as percentages rather than decimals) might be helpful.

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

The following color gradient can be used as a general scale interpretation of the BLEU score:

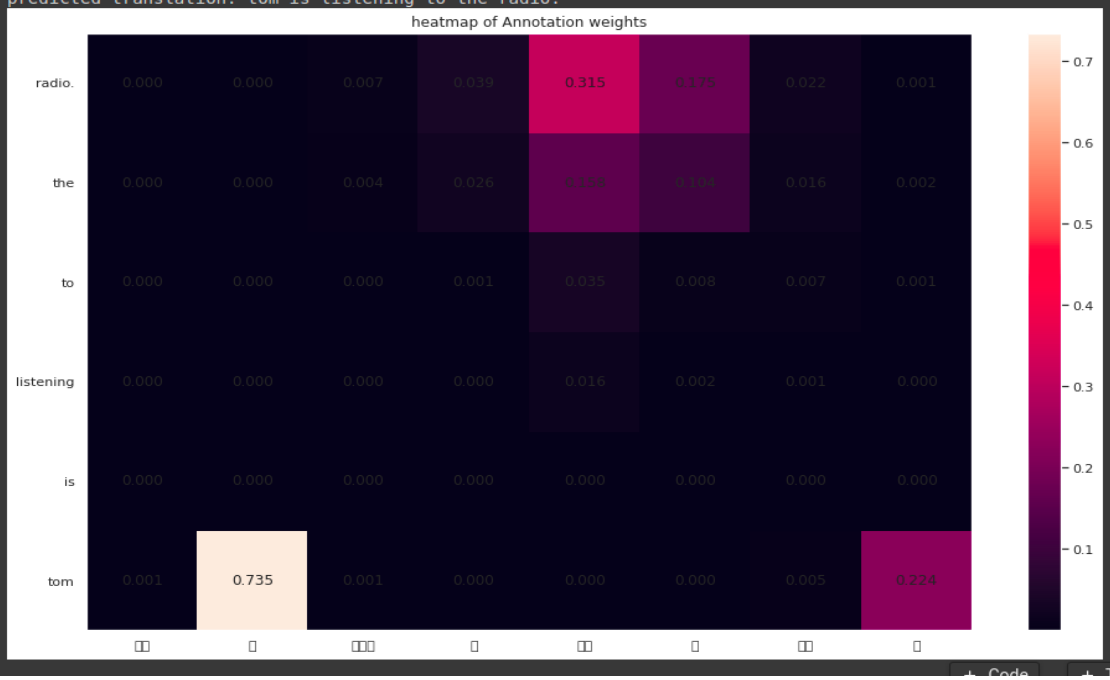
Fig 10 : BLEU SCORE Table

Our BLEU score after the last epoch was 45.26. By the table above we can say that our model has worked quite well.

5.2.2 Predicted Translation and Heatmap Annotations:

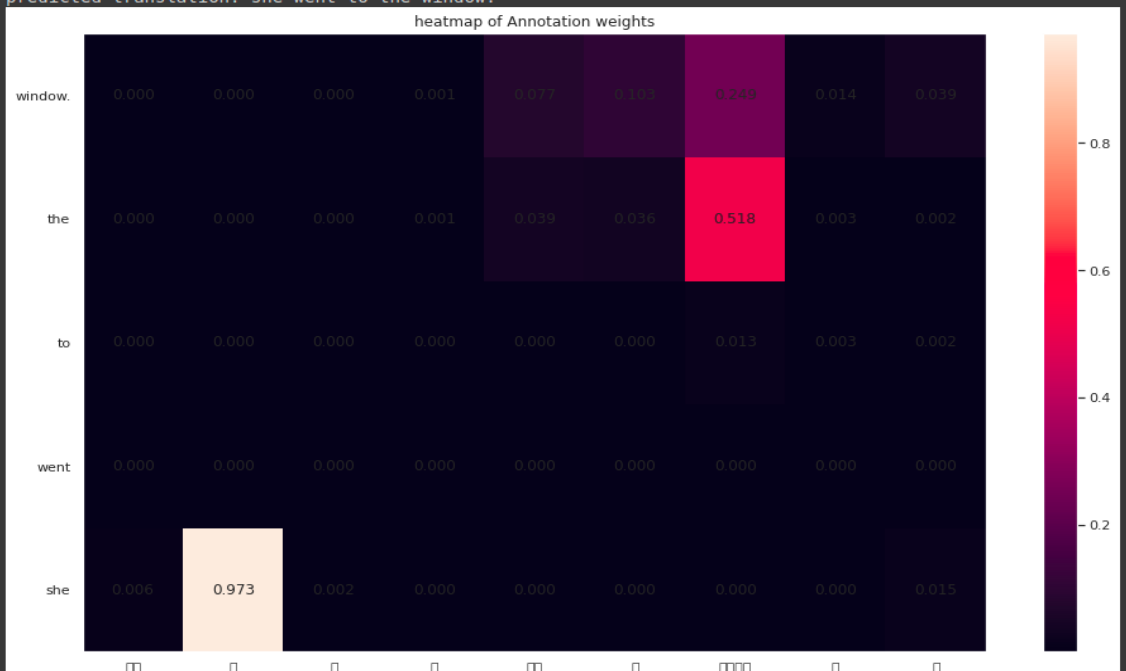
```
model.test()

input sentence: トム は ラジオ を 聞い て いる 。
actual translation: tom is listening to the radio.
predicted translation: tom is listening to the radio.
```



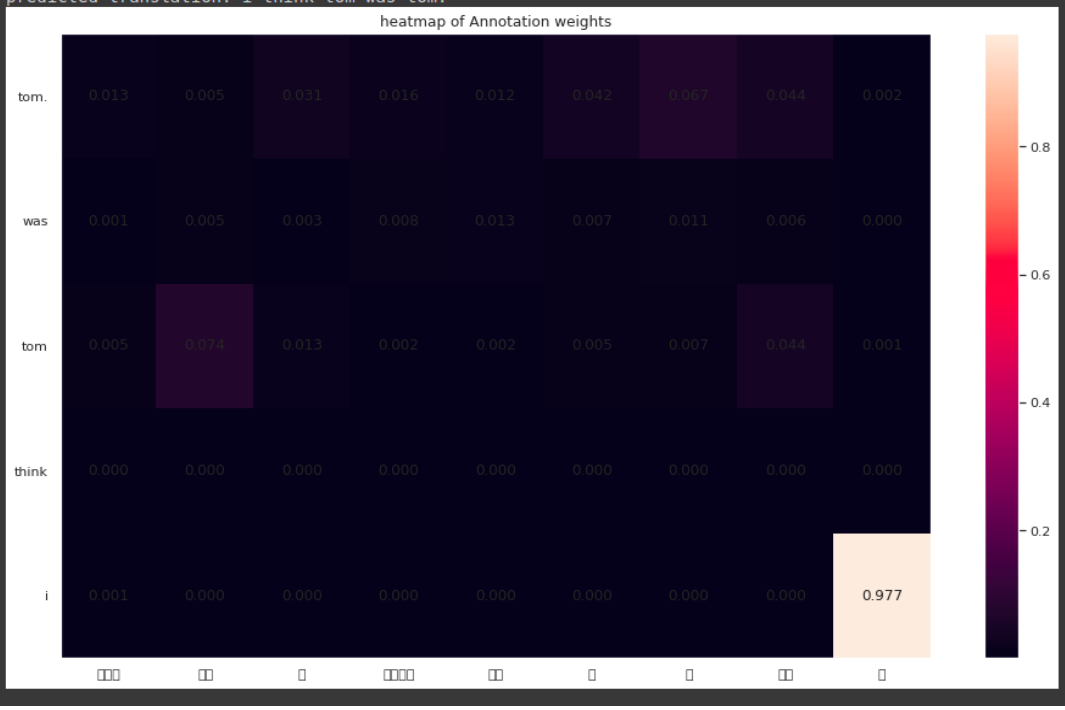

```
model.test()

input sentence: 彼女 は 彼 を いす に 縛り付け た 。
actual translation: she tied him to the chair.
predicted translation: she went to the window.
```



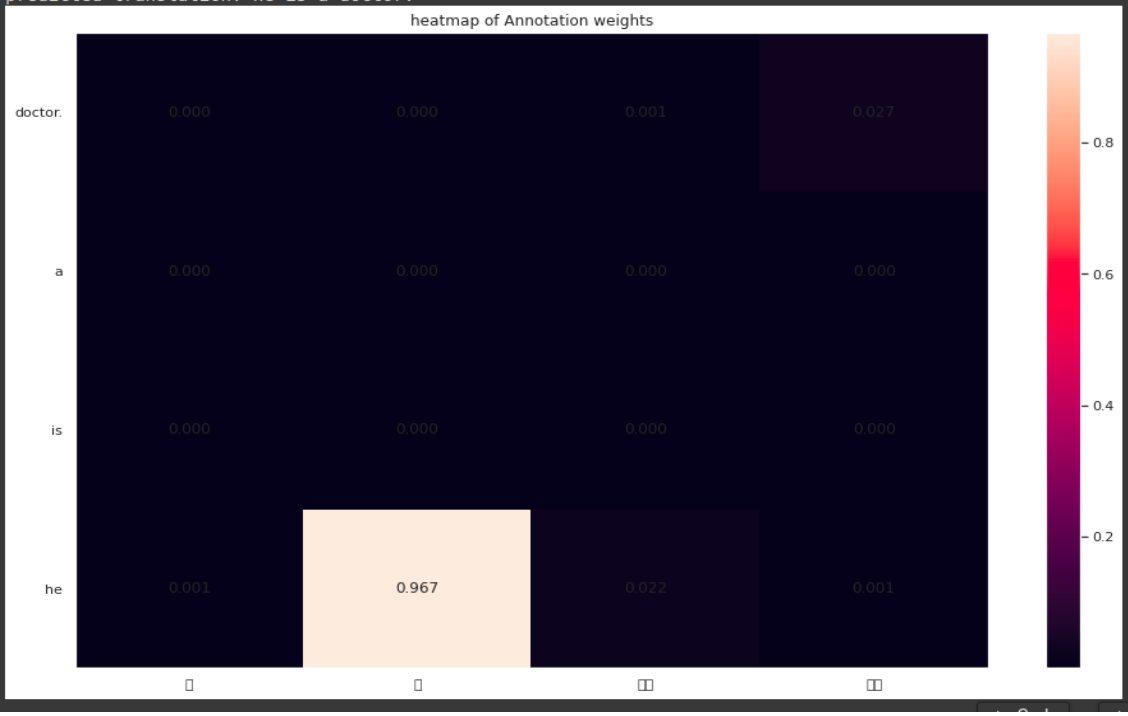
```
model.test()
```

```
input sentence: たしか トム の アイデア だっ た と 思う 。
actual translation: i assume it was tom s idea.
predicted translation: i think tom was tom.
```



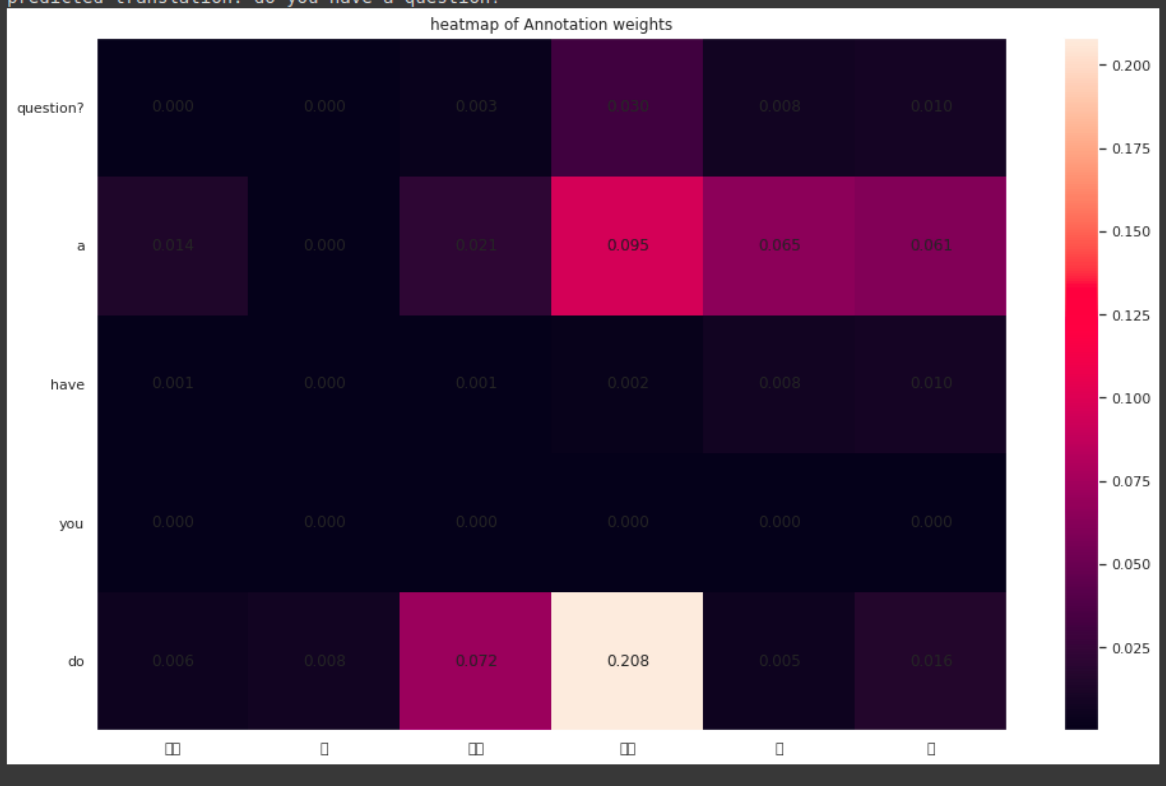

```
model.test('彼は医者です')
```

```
input sentence: 彼 は 医者 です
predicted translation: he is a doctor.
```



```
model.test()
```

```
input sentence: 質問 は あり ます か ？
actual translation: do you have any questions?
predicted translation: do you have a question?
```



6. Conclusion

As the amount of content that needs to be localised expands exponentially, neural machine translation will be utilised at a higher rate in the future.

The COVID-19 situation has expedited many firms' digital transformations, resulting in a greater demand for translation services. Simultaneously, material must be more targeted and diverse. Machine Translation will be employed for sections of the content, with or without human translation oversight, as a result of these market conditions.

Brands looking to localise their content are turning to cloud-based translation management solutions as a new frontier of innovation. The systems use advanced technologies like translation memory and machine translation to boost the speed and quality of translation services, allowing businesses to reach out to global customers with unparalleled efficiency and precision.

With the localisation of the data is possible using cloud, we might be able to use the Cloud Based GPUs provided by various companies, which will help us in increasing the efficiency of the translation process. With the recent advent of much powerful GPUs, we might afford better results without even using cloud based GPUs, but the memory output of the process will be quite high.

Our struggle to overcome language barriers has decreased a little bit but it didn’t end. This proves to an extent how technology can never replace human emotions and intelligence, even though we train the best of models with the best of inputs fed in. Never-the-less, we have found out that the more the epochs in our model, the better will be our bleu score, and for that we will need higher levels of Cloud Gpu. We have worked with the free GPU provided to us by the google colaboratory.

7. References

S.No	Title	Publication/Conference	Year	Link
1.	Attention-based Multimodal Neural Machine Translation By Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh , Chris Dyer	Association for Computational Linguistics	2016	Attention-based Multimodal Neural Machine Translation
2.	Neural Machine Translation by Jointly Learning to Align and Translate By Dzmitry Bahdanau Jacobs , KyungHyun Cho, Yoshua Bengio	ICLR	2015	https://arxiv.org/pdf/1409.0473.pdf
3.	Contrastive Attention Mechanism for Abstractive Sentence Summarization By Xiangyu Duan , Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo , Yue Zhang	Association for Computational Linguistics	2019	https://arxiv.org/pdf/1910.13114.pdf
4.	Abstractive Summarization Using Attentive Neural Techniques By Jacob Krantz & Jugal Kalita	Researchgate	2018	https://arxiv.org/pdf/1810.08838.pdf
5.	Sequence to Sequence Learning with Neural Networks By Ilya Sutskever, Oriol Vinyals, Quoc V. Le	NeurIPS	2014	https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

6.	Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling By Hasim Sak, Andrew Senior, Francoise Beaufays	INTERSPEECH	2014	https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf
7.	Attention in Natural Language Processing By Andrea Galassi , Marco Lippi , and Paolo Torroni	IEEE	2021	https://arxiv.org/ftp/arxiv/papers/1902/1902.02181.pdf
8.	Selective Encoding for Abstractive Sentence Summarization By Qingyu Zhou , Nan Yang, Furu Wei, Ming Zhou	Association for Computational Linguistics	2017	https://arxiv.org/pdf/1704.07073.pdf
9.	Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network By Alex Sherstinsky	Elsevier journal	2020	https://arxiv.org/pdf/1808.03314.pdf
10.	A Critical Review of Recurrent Neural Networks for Sequence Learning By Zachary C. Lipton, John Berkowitz, Charles Elkan	Researchgate	2015	https://arxiv.org/pdf/1506.00019.pdf