CS779 Competition: Machine Translation System for India

Harsh Bihany
210406
{harshb21}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

In this report, I present my approach to the Machine Translation Competition. I explored various machine translation models, starting from a basic Seq2Seq architecture and progressively incorporating global context vectors, attention mechanisms, and bi-directionality. Our attention-based models significantly outperformed the vanilla Seq2Seq models, showcasing the power of the attention mechanism in capturing linguistic nuances. Our experiments were primarily centered around translating English sentences to their corresponding Indic counterparts (Bengali, Gujarati, Hindi, Kannada, Malayalam, Tamil, and Telgu). Our best model achieved a **BLEU score** of 0.054, placing me 9^{th} on the final leaderboard.

1 Competition Result

Codalab Username: H_210406
Final leaderboard rank on the test set: 9
charF++ Score wrt to the final rank: 0.215
ROGUE Score wrt to the final rank: 0.244
BLEU Score wrt to the final rank: 0.054

2 Problem Description

Machine Translation System for India (Phase 1), is the projection of a popular *Natural Language Processing* sub-domain of Machine Translation, i.e., translating text from one language to another using computational algorithms, in the Indian context.

The objective of the machine translation task was to develop models of capable of translating *English* sentences to their corresponding *Indic* counterparts (the languages in the Indian context were: Bengali, Gujarati, Hindi, Kannada, Malayalam, Tamil and Telgu).

Machine translation is an inherently tough task, because of *morphological complexity* present in languages, *variation of scripts* in languages and *semantic preservation* of text amongst other hurdles.

3 Data Analysis

Let's begin by some elementary introduction to the training dataset: `train_data1.json`.

3.1 Dataset size

We have three *Indo-Aryan* languages (Bengali, Gujarati, Hindi), and four *Dravidian* languages (Kannada, Malayalam, Tamil, Telgu).

Language-pair	Number of sentence pairs
English-Bengali	68849
English-Gujarati	47482
English-Hindi	80797
English-Kannada	46795
English-Malayalam	54057
English-Tamil	58361
English-Telgu	44905

Table 1: Size of train dataset

We have the most number of English-Hindi pairs, which is also somewhat expected, because of the vast prevalence of Hindi.

3.2 Vocabulary size

Following are vocabulary size (`tokenization` used), for different languages. We have used `spacy`'s `en_core_web_sm` for English purposes, and we have used `indic_nlp`'s `trivial_tokenize` for different languages. following are the vocabulary sizes of various languages.

Language	Vocabulary size
English (non-processed)	106178
English (processed)	79321
Bengali	96952
Gujarati	84045
Hindi	75579
Kannada	98717
Malayalam	137674
Tamil	131617
Telgu	90330

Table 2: Vocab sizes

Processing of English corpus includes, removing capitalization and abnormal characters, and other common `regex` operations. Others were normalized using `indic_nlp`'s normalizer.

We can infer, that the tokenizers for the Dravidian langauges are not properly curated, because of the large vocab size, and possibly due to the nature of the language.

3.3 Length distribution

Plots shown are made after removing the absolute outliers.

3.3.1 Hindi

Consider the English-Hindi length of sentence distribution (*the length of sentence is the number of tokens it is divided into*; previous tokenizations hold)

	English	**Hindi**
Average	20.66	22.69
95-**percentile**	40	45
Maximum	296	333

Table 3: Statistics regarding length of English-Hindi pairs of sentences

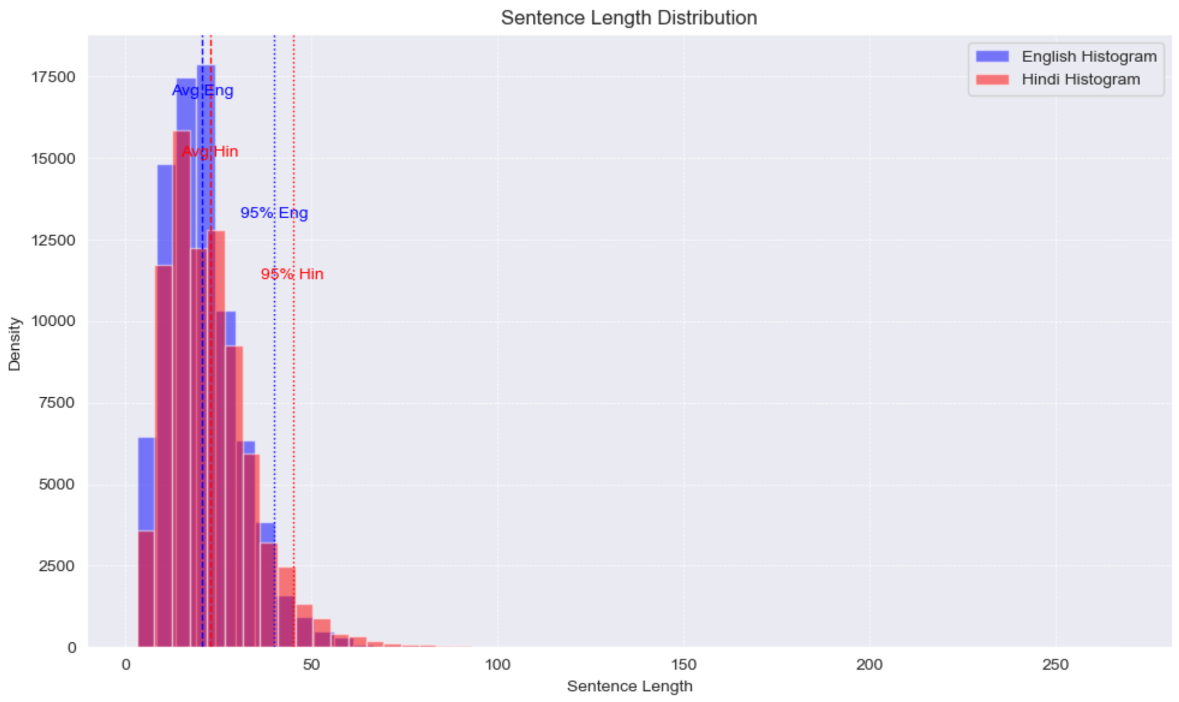

Figure 1: English-Hindi length of sentences

3.3.2 Tamil

	English	**Tamil**
Average	19.93	15.54
95-percentile	37	30
Maximum	2445	100

Table 4: Statistics regarding length of English-Tamil pairs of sentences

3.4 Word Cloud

Word clouds Heimerl et al. [2014] are visual representations of text data where the size of each word indicates its frequency or importance, with frequently occurring terms displayed prominently and larger than less common terms. They provide a quick way to discern the most prominent terms in a body of text.

Figure 3 is the Word Cloud for the entire English corpus

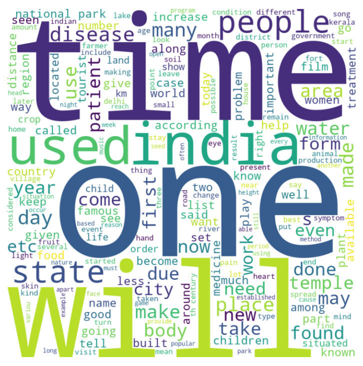

Figure 3: WordCloud for English

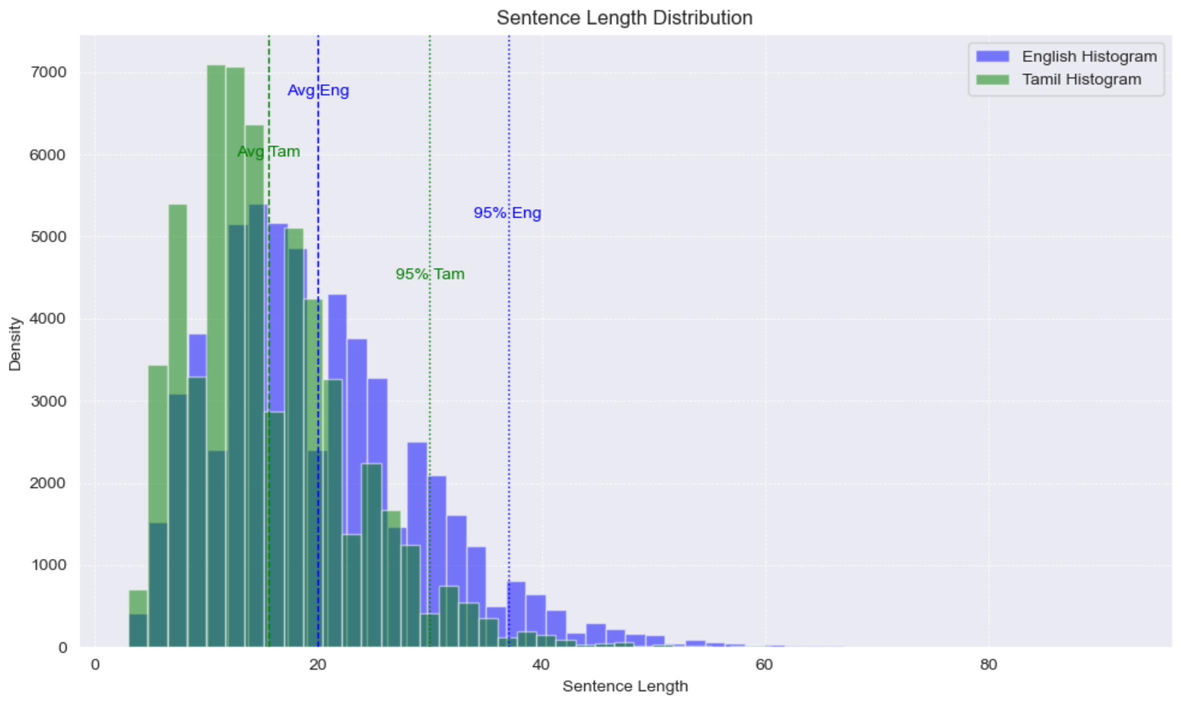

Figure 2: English-Tamil length of sentences

4 Model Description

4.1 Model trial 1: A basic Seq2Seq architecture

I started with a basic Seq2Seq architecture based out of **GRU** Chung et al. [2014], for machine translation, between the English script and a Indic language.

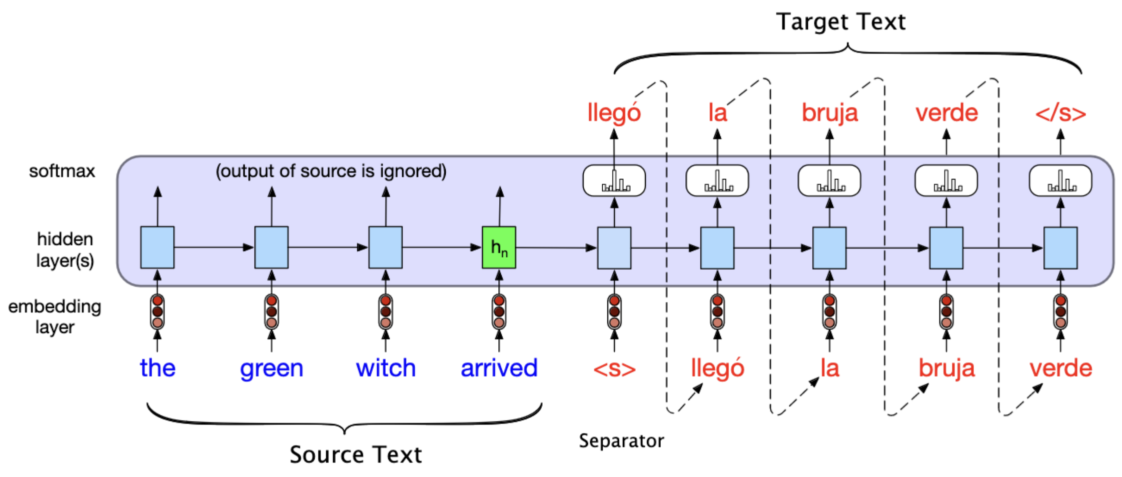

Figure 4: Basic Seq2Seq Architecture[1]

The observed translations showed results, on which a lot of focus was on terminating punctuations (them being repeated in the translation), and some words which were just being repeated again and again. This perhaps indicates at the fact that the model was incapable of holding information for longer, and had memory localization.

[1]Speech and Language Processing-Daniel Jurafsky, James H. Martin

4.2 Model trial 2: Global context vector

I scaled up my model size. I observed that `colab` GPU's were not used to their full potential in the previous run of things. I scaled up the model dimensions, keeping the dataloaders same for both. Further, in my previous Seq2Seq architecture, as illustrated by several papers and research, the single `context vector` acts as *bottleneck* between the encoder and decoder.

The model this time around, had the final hidden output of the GRU encoder, feeded into each node of the decoder.

There was a slight improvement in the results, compared to the previous turn of events. However still, the results reflected memory localizations.

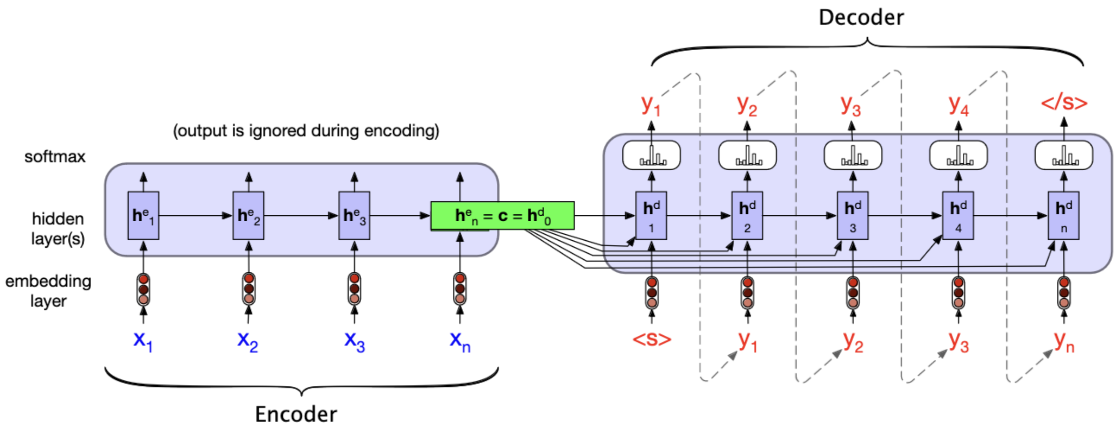
Decoder
y_1 y_2 y_3 y_4 </s>
(output is ignored during encoding)
softmax
hidden layer(s)
embedding layer
h^e_1 h^e_2 h^e_3 $h^e_n = c = h^d_0$
h^d_1 h^d_2 h^d_3 h^d_4 h^d_n
x_1 x_2 x_3 x_n
<s> y_1 y_2 y_3 y_n
Encoder

Figure 5: Modified Seq2Seq Architecture

4.3 Model trial 3: Attention

Taking inspiration from the `Pytorch`'s official examples: **NLP from Scratch**, I implemented my version of attention, modifying my previous models, by incorporating in them *Self-Attention* Bahdanau et al. [2016].

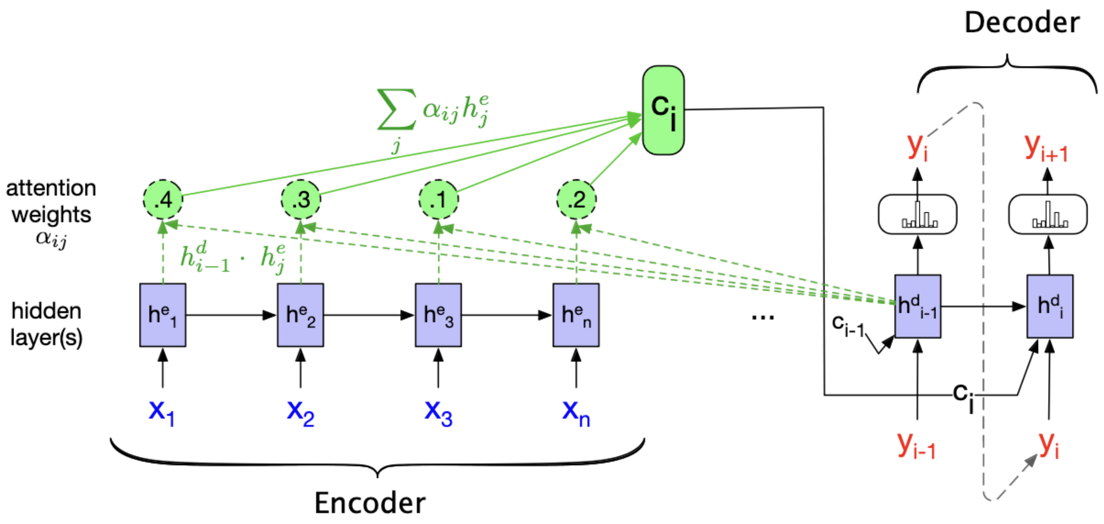

Figure 6: Attention based Seq2Seq Architecture

Compared to the previous models, there was a staggering difference. Even a visual one. Though the training was slow, the translated sentences had some semblance of linguistic text, unlike the previous models, which produced near to gibberish. This is also my current submission.

The model parameters were as follows, the size of the GRU `hidden dimension` was 128, and the `embedding dimension` was 256. The `batch size` was 16, and model was trained over 10 epochs over the training data.

4.4 Model trial 4: Introducing Bi-directionality

The last modification was a small one, in which the forward GRU's of the encoder, were made to be bi-directional. This required modifications in how Attention and decoder handled encoder-outputs.

I did achieve better results for the `chF++` score, however, there was a slight dip in the `BLEU` score. Though this is not expected, this anomaly can be attributed to irregular data pruning, held this time around.

5 Experiments

The Data-preprocessing was borrowed from the **Indic-NLP** Kunchukuttan [2020], documentation over Indian language, tokenization and normalization.

For the English-language, I however chose to stick with `spacy`'s Honnibal and Montani [2017], built-in English support.

The optimizer used was the `Adam` optimizer, with a `learning_rate` of 0.001. I also chose to use the `NLLLoss` (Negative Log Likelihood) loss, because the model gave out *logits*, rather than raw outputs.

Trying different models, and training them, provided little time for rigorous hyper-parameter tuning, thus I chose to stick with most default hyper-parameters, including the optimizer learning rate, decay rate etc. One hyper-parameter which was important in the training process, was the ignoring of the `PAD_IDX` for loss calculation.

The major experimentations were with playing with the **GRU-based Seq2Seq model**, and applying add-ons to it going forward.

6 Results

Model	charF++ score	ROGUE score	BLEU score	Total score	Rank
4.1	0.095	0.192	0.013	0.300	-
4.2	0.113	0.228	0.025	0.366	-
4.3	**0.213**	**0.243**	**0.054**	**0.510**	**7**
4.4	0.216	0.242	0.047	0.504	9

Table 5: Results for various models on the dev set

The attention mechanism is a thing of beauty, and it depicted rightly so here. Compared to baseline vanilla Seq2Seq models, it had shown major improvements.

A bit of ambiguity was the almost similar performances of the bidirectional attention, and attention on the forward seq2seq model. A plausible reason, could be the size of model. The model size is seemingly miniature when compared to vocab size. Hence perhaps, the added information brought in the the bi-directionality of the model, does not make significant improvements.

7 Error Analysis

Attention based models outperformed vanilla-RNN based Seq2Seq models by a long shot. This can be attributed to the way **attention** works. It allows the model to focus on specific parts of the input sequence when producing an output, much like how humans pay attention to particular parts of input when understanding or translating a sentence. In essence, attention provides a weighted context to the model, enabling it to produce more accurate and coherent outputs by considering which parts of the input are most relevant at each step of the output generation.

Attention map

Attention maps, often visualized as heatmaps, are graphical representations of the attention weights in models that use the attention mechanism, such as in many sequence-to-sequence tasks. These maps provide insights into *which parts of the input sequence the model focuses on when producing each part of the output sequence.* In the context of machine translation, for instance, an attention map can show which words in the source sentence were most influential in determining each word in the translated sentence. Brighter or darker regions in the heatmap indicate higher attention weights, revealing the model's focus and offering interpretability into its decision-making process.

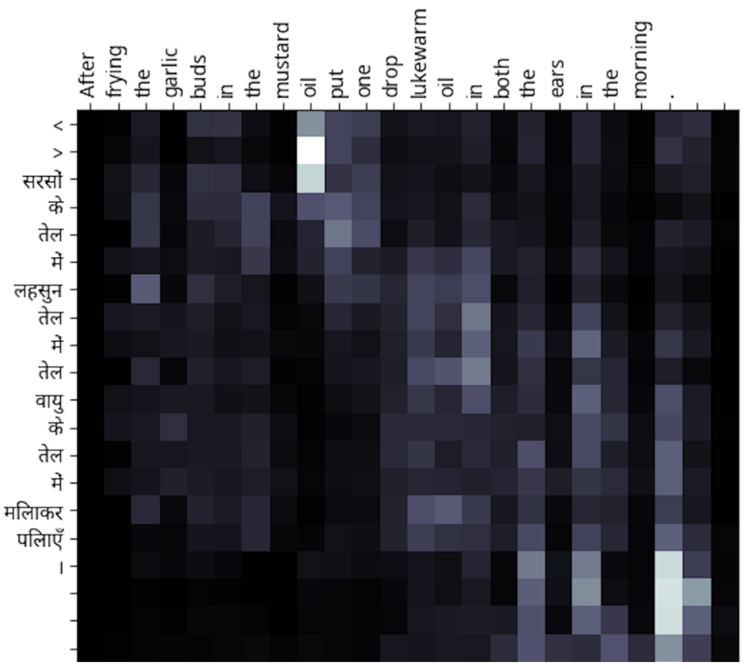

Figure 7: Attention heat map for "English-Hindi", pair number 567

We can observe, a general trend that the model does learns to produce some relevant words and attends to some words, however coherency is still an issue, and there is a massive scope for improvement.

No model can be perfect. Specially in the case of *machine translation.* A primary reason is conflicting opinions of language connoisseur on what the quality of languages mean. Even if we hold the **present metrics** to be the gold standard of language coherency, languages over years of cultural mixes, have become so complex, it is not possible for a human, let alone a machine to understand the hidden intricacies of it.

8 Conclusion

We have established that recurrent neural networks, which are *Turing complete* Chung and Sieglemann [2021], fail in practical settings to capture the essence of language completely, without external aid.

Attention mechanism which is a massive improvement over its baselines. Hence it is highly recommended that one begins by internalizing the attention mechanism and how it works, after the basics of the *encoder-decoder* task is understood. One important note to take from the assessment apart from the obvious learning into such model architectures and how they work, was to maintain the readability of the code for future reference and access. It proved important to redo and enhance existing code base time and again.

Future Directions:
- Using **sub-word tokenization**
- **Transformers** as a machine translation model Vaswani et al. [2023] {I had started to train my transformer model, but training could not finish training before the deadline.}

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Stephen Chung and Hava Sieglemann. Turing completeness of bounded-precision recurrent neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=IWJ9jvXAoVQ`.

Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. Word cloud explorer: Text analytics based on word clouds. In *2014 47th Hawaii International Conference on System Sciences*, pages 1833–1842, 2014. doi: 10.1109/HICSS.2014.231.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Anoop Kunchukuttan. The IndicNLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.