# Machine Translation with Attention Mechanism

**Pragathi Gopishetty, Ranjith Reddy Mada**

## Abstract

An attentional mechanism has recently been used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. However, little research has been conducted into useful architectures for attention-based NMT. The attention mechanism is appealing for neural machine translation because it can dynamically encode a source sentence by generating an alignment between target and source words.

In this paper, We show the effectiveness of approaches on NMT translation tasks to convert from English to Telugu, and we also used predefined hugging face transformer models to determine whether accuracy increases or decreases.

## 1. Introduction

Neural Machine Translation (NMT) has had a lot of success with machine translation tasks (Bahdanau et al., 2015; Sutskever et al., 2015). Under the Encode-Decode framework, it typically uses a recurrent neural network to encode a source sentence into context vectors before generating a token-by-token translation from the target vocabulary. Among the various types of NMT, attention-based NMT, which is the subject of this paper, is gaining popularity in the community (Bahdanau et al., 2015; Luong et al., 2015). One of its advantages is that it can use an attention mechanism to dynamically use the encoded context, allowing for the use of fewer hidden layers while still maintaining high translation performance.

In large-scale translation tasks such as English to Telugu, Neural Machine Translation (NMT) achieved state-of-the-art results. NMT is appealing because it requires little domain knowledge and is conceptually straightforward. The model developed by Luong et al. (2015) reads through all of the source words until it reaches the end-of-sentence symbol <eos>. It then begins to emit one target word at a time. NMT is typically a large neural network that has been trained from beginning to end and can generalise well to very long word sequences. As a result, unlike standard MT, NMT does not require the model to explicitly store massive phrase tables and language models; thus, NMT has a small memory footprint. Finally, unlike standard MT decoders, NMT decoders are simple to implement (Koehn et al., 2003).

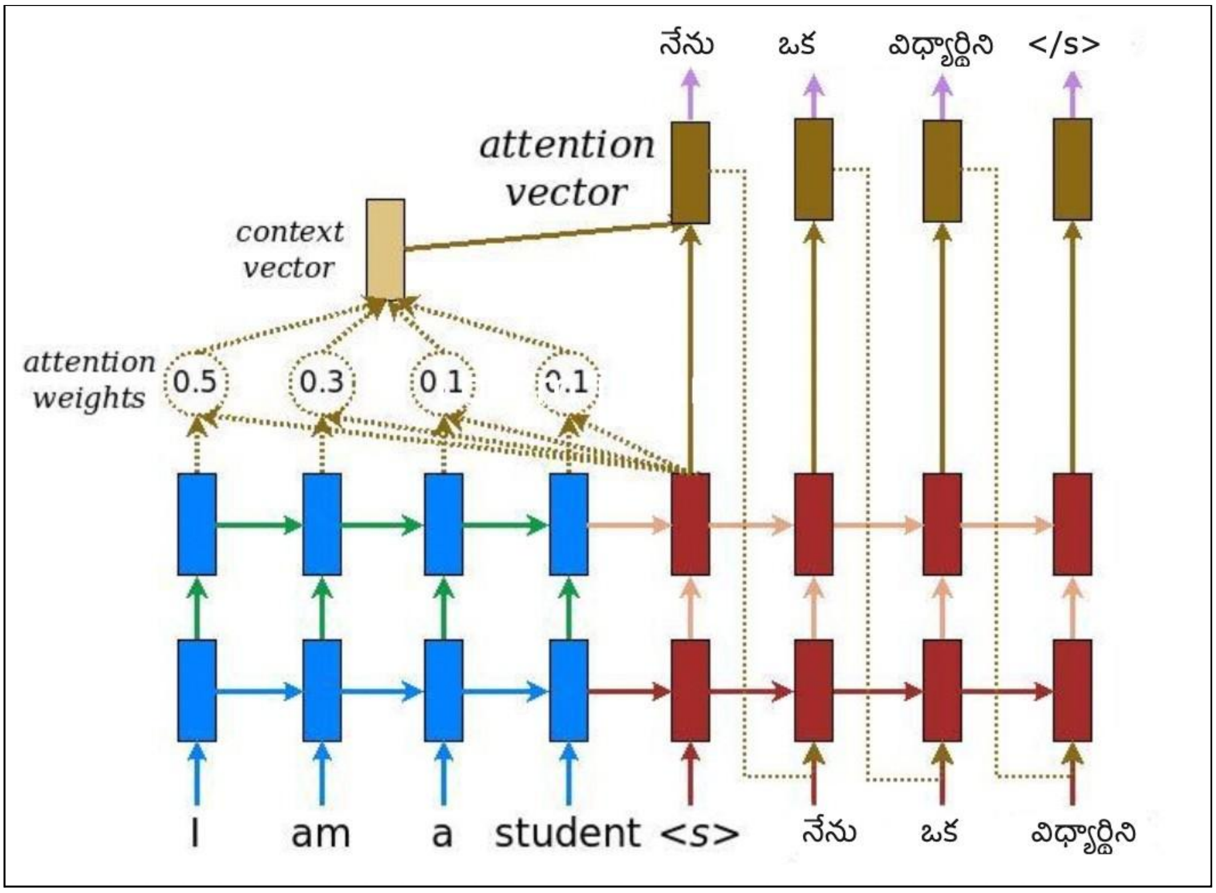


**Figure 1: Encode Decoder Architecture**

## 2. Neural Machine Translation

A neural machine translation system is a neural network that models the conditional probability p(y|x) of translating a source sentence, x1,...,xn, to a target sentence, y1,...,ym. A basic form of NMT consists of two components: (a) an encoder that computes a representations for each source sentence and (b) a decoder that generates one target word at a time and thus decomposes the conditional probability as follows:

$$\log p(y|x) = \sum_{j=1}^{m} \log p\left(y_j | y_{<j}, \boldsymbol{s}\right)$$

A recurrent neural network (RNN) architecture is a natural choice for modelling such a decomposition in the decoder (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Jean et al., 2015). However, the RNN architectures used for the decoder and how the encoder computes the source sentence representations differ.

## 2.1 Attention-based Models

The encoder memorises a complete sentence after reading it. The activation layer is where it is kept. It is used by the decoder to generate the translated target sentence. This mechanism performs better than the other models for short sentences, but it degrades performance for longer sentences of 30 or 40 words. Longer sentences can be handled using an encoder decoder architecture with attention. The attention mechanism notices dependencies in the input or output regardless of how far apart they are. By looking at parts of the text at a time, it translates like human-generated sentences. During the sentence translation process, the approach determines how much attention should be paid to a specific word. The encoder takes the input vector

$X1$, $X2,\dots,Xt$. X1,X2,…,Xt and generates the attention vector $h1,h2,\dots,ht$. h1,h2,…,ht. The context vector $Ci$. $Ci$ is produced by concatenating these vectors for each time step of the input. The decoder generates the target word using context vector, hidden state, and previously predicted word.

Global and local attention-based models are the two broad categories in which we categorise our various attention-based models. These classes differ in terms of whether "attention" is paid to all source positions or just a few of them.

## 2.2 Global Attention

Global attention is a recurrent neural network extension of the attentional encoder-decoder model. Although it was designed for machine translation, it can also be used for caption generation and text summarization, as well as sequence prediction tasks in general.

For an encoder-decoder network that predicts one time step given an input sequence, we can divide the calculation of global attention into the following computation steps.

- **Problem**. The input sequence is provided as input to the encoder (X).
- **Encoding**. The encoder RNN encodes the input sequence and outputs a sequence of the same length (hs).

- **Decoding**. The decoder interprets the encoding and outputs a target decoding (ht).
- **Alignment**. Each encoded time step is scored using the target decoding, then the scores are normalised using a softmax function. Four different scoring functions are proposed:
    - **dot**: the dot product between target decoding and source encoding.
    - **general**: the dot product between target decoding and the weighted source encoding.
    - **concat**: a neural network processing of the concatenated source encoding and target decoding.
    - **location**: a softmax of the weighted target decoding.
- **Context Vector**. The alignment weights are applied to the source encoding by calculating the weighted sum to result in the context vector.
- **Final Decoding**. The context vector and the target decoding are concatenated, weighed, and transferred using a tanh function.

The final decoding is run through a softmax to estimate the likelihood of the next word in the sequence over the output vocabulary.

The diagram below depicts the data flow when calculating global attention at a high level.

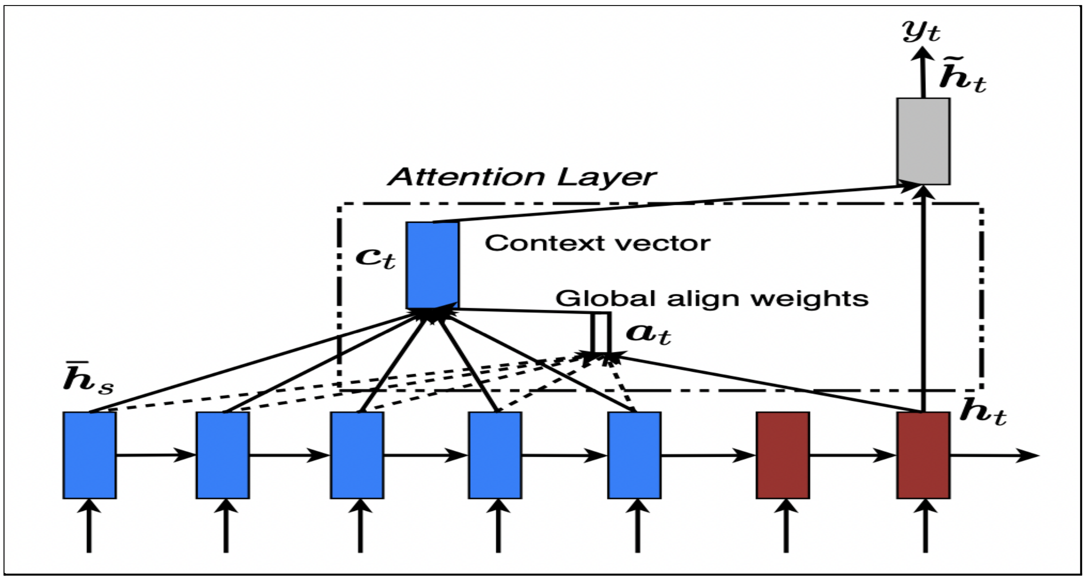


**Figure 2.2: Global Attention Architecture**

## 2.3 Local Attention

The disadvantage of global attention is that it must pay attention to all words on the source side for each target word, which is costly and may make it impractical to translate longer sequences, such as paragraphs or documents. To make up for this shortcoming, we propose a local attentional mechanism that selects a small subset of source positions per target word to focus on.

The tradeoff between the soft and hard attentional models proposed by Xu et al. (2015) to tackle the image caption generation task inspired this model. Soft attention, in their work, refers to a global attention approach in which weights are "softly" applied to all patches in the source image. Hard attention, on the other hand, chooses one patch of the image at a time to focus on. The hard attention model, while less expensive at inference time, is non-differentiable and requires more complicated techniques to train, such as variance reduction or reinforcement learning.

Our local attention mechanism is distinct in that it selectively focuses on a small window of context. This method has the advantage of avoiding the costly computations associated with soft attention while also being easier to train than the hard attention method. In more detail, at time t, the model generates an aligned position pt for each target word. The context vector ct is then calculated as a weighted average of the set of hidden source states within the window [ptD,pt+D]; Disempiricallyselected. 8 The local alignment vector is now fixed-dimensional, i.e. R2D+1, in contrast to the global approach. We consider two model variants. *Monotonic* alignment (**local-m**) *Predictive* alignment (**local-p**)

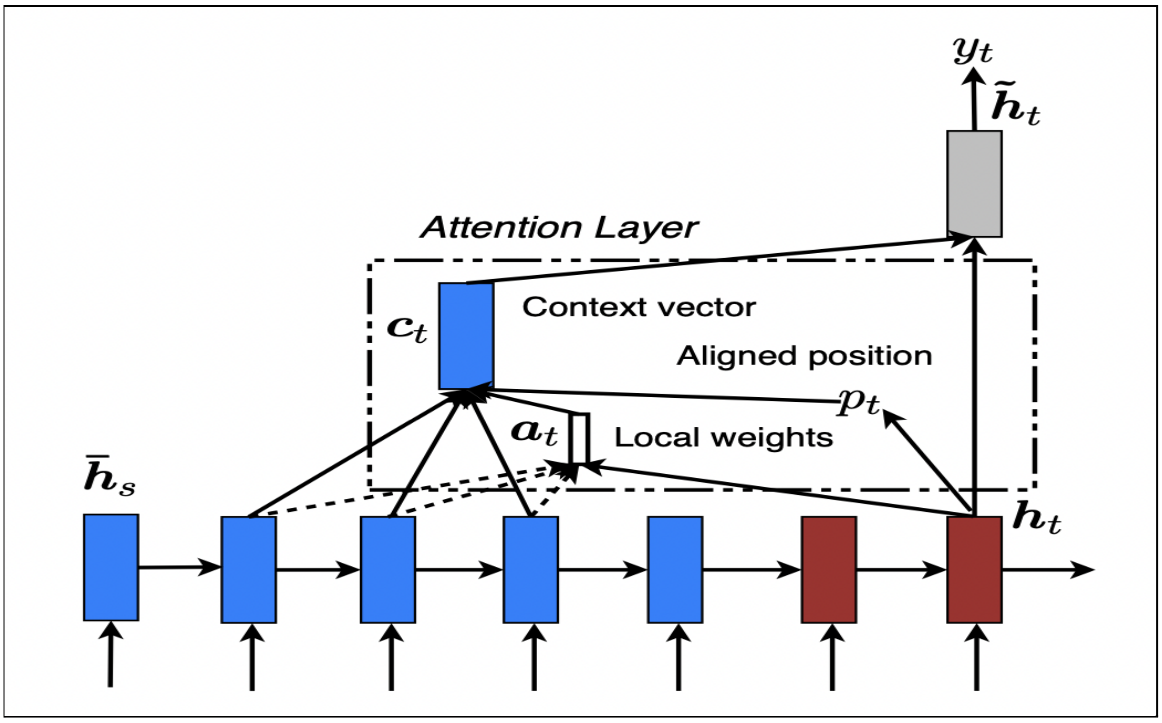


**Figure 2.3: Local Attention Architecture**

## 2.4 The attention head

The decoder employs attention to focus on specific parts of the input sequence. For each example, the attention takes a sequence of vectors as input and returns an "attention" vector. This is a layer that is similar to a layer. However, the attention layer uses a weighted average instead of GlobalAveragePoling1D.

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \qquad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \qquad \text{[Context vector]}$$

Where:

- $s$ is the encoder index.
- $t$ is the decoder index.
- $\alpha_{ts}$ is the attention weights.
- $h_s$ is the sequence of encoder outputs being attended to (the attention "key" and "value" in transformer terminology).
- $h_t$ is the the decoder state attending to the sequence (the attention "query" in transformer terminology).
- $c_t$ is the resulting context vector.
- $a_t$ is the final output combining the "context" and "query".

This layer takes 3 inputs:

- The query: This will be generated by the decoder, later.
- The value: This Will be the output of the encoder.
- The mask: To exclude the padding, example_tokens != 0

The vectorized implementation of the attention layer lets you pass a batch of sequences of query vectors and a batch of sequence of value vectors. The result is:

1. A batch of sequences of result vectors the size of the queries.
2. A batch attention maps, with size (query_length, value_length).

## 3. Experiments

We examine the effectiveness of our models on NMT translation tasks in both directions from English to Telugu.

### 3.1 Training Details

Our models are built using 93K sentence pairs as training data (93K English words, 93K Telugu words). After that, data was preprocessed. Tokenizer is used to apply the tokenization technique later.

### 3.1.1 The encoder/decoder model

An overview of the model is shown in the diagram below. To predict the next word, the decoder's output is combined with a weighted sum over the encoded input at each time step. The diagram and formulas are from Luong's paper.

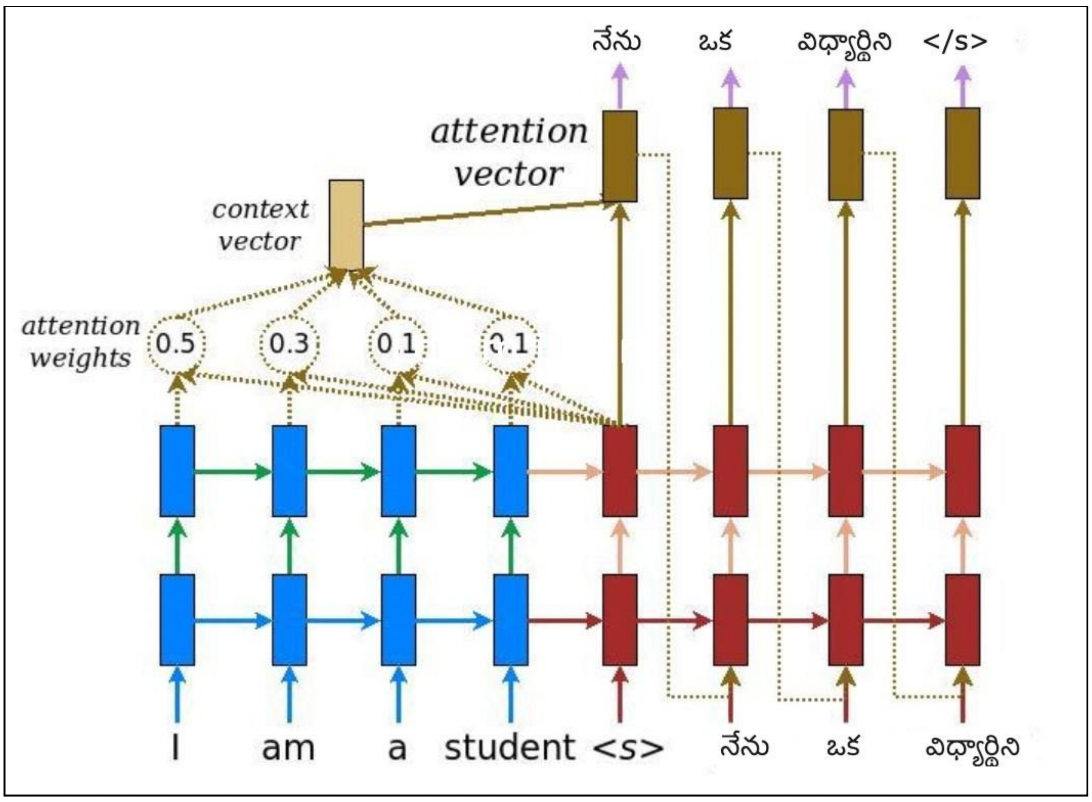


**Figure 3.1.1: Attention model architecture**

### 3.1.2 Encoder

1. Takes a list of token IDs (from input_text_processor).
2. Look up an embedding vector for each token (Using a layers.Embedding).
3. Processes the embeddings into a new sequence (Using a layers.GRU).
4. Returns:

   The processed sequence. This will be passed to the attention head.

   The internal state. This will be used to initialise the decoder

### 3.1.3 Decoder with attention

The decoder's job is to generate predictions for the next output token.

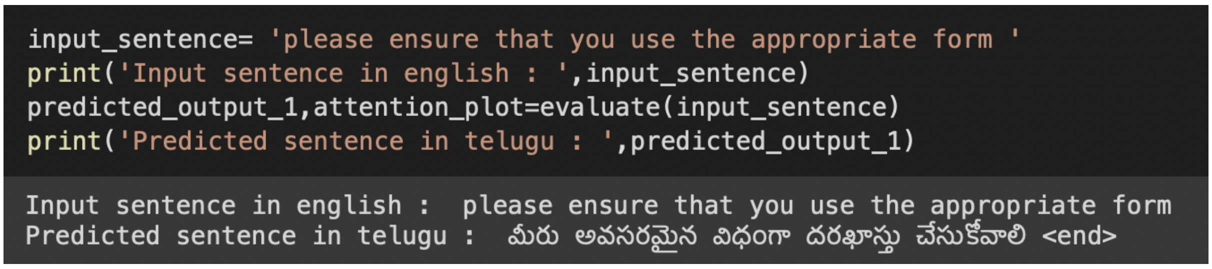
1. The decoder receives the complete encoder output. It uses an RNN to keep track of what it has generated so far.
2. It uses its RNN output as the query to the attention over the encoder's output, producing the context vector.
3. It combines the RNN output and the context vector to generate the "attention vector".
4. It generates logit predictions for the next token based on the "attention vector".

The train_step method, added below, handles the remaining steps except for actually running the decoder. The model was then built using 25 epochs.

## 4.2 English-Telugu Results

1. **Encoder Decoder with Attention:**

   The following is the output for the english to telugu conversion using encoder decoder with attention model.

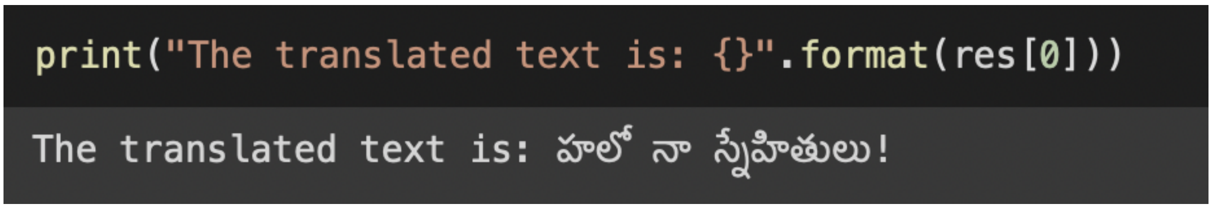


```
input_sentence= 'please ensure that you use the appropriate form '
print('Input sentence in english : ',input_sentence)
predicted_output_1,attention_plot=evaluate(input_sentence)
print('Predicted sentence in telugu : ',predicted_output_1)
```

```
Input sentence in english :  please ensure that you use the appropriate form
Predicted sentence in telugu :  మీరు అవసరమైన విధంగా దరఖాస్తు చేసుకోవాలి <end>
```

**Figure 4.2.1: Output for Encoder Decoder**

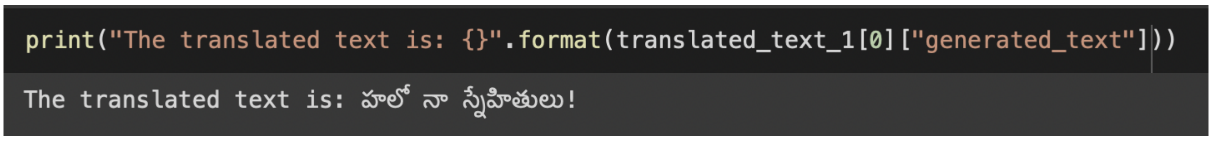
2. **facebook/mbart-large-50-one-to-many-mmt:**
   - This model is a fine-tuned version of the mBART-large-50 checkpoint.
   - It has been fine-tuned for machine translation into multiple languages.
   - It was introduced in the Multilingual Translation with Extensible Multilingual Pretraining and Fine Tuning paper.
   - The model can convert between English and 49 other languages. The target language id is forced as the first generated token when translating into a target language.



```
print("The translated text is: {}".format(res[0]))
The translated text is: హలో నా స్నేహితులు!
```

**Figure 4.2.2: Output for predefined model**

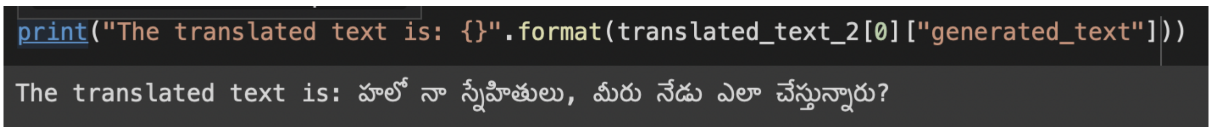
3. **Helsinki-NLP/opus-mt-en-dra:**
    - This model can translate English to Dravidian languages.
    - Pre-processing: normalisation + SentencePiece.
    - A sentence initial language token is required in the form of >>id<< (id = valid target language ID).
    - Bleu score - 7.1



```
print("The translated text is: {}".format(translated_text_1[0]["generated_text"]))
The translated text is: హలో నా స్నేహితులు!
```

**Figure 4.2.3: Output for predefined model**

4. **Helsinki-NLP/opus-mt-en-mul**
    - The mode can translate from English to a variety of other languages.
    - Pre-processing: Normalisation + SentencePiece
    - A sentence initial language token is required in the form of >>id<< (id = valid target language ID).
    - BLEU score - 4.7

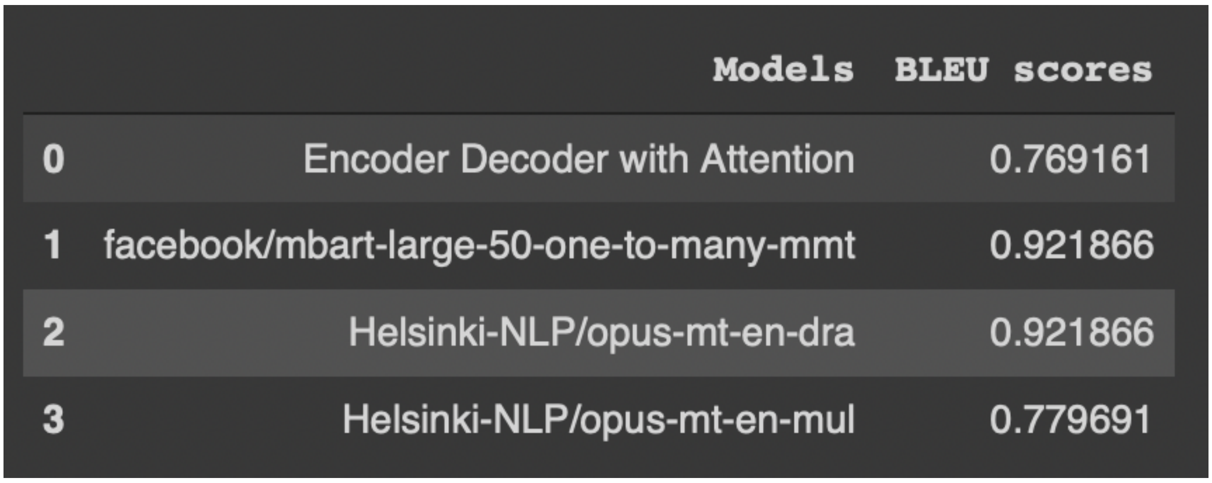


```
print("The translated text is: {}".format(translated_text_2[0]["generated_text"]))
The translated text is: హలో నా స్నేహితులు, మీరు నేడు ఎలా చేస్తున్నారు?
```

**Figure 4.2.4: Output for predefined model**

**BLEU scores:**

From the below analysis we can observe the BLEU scores for all the four models and we can figure out that predefined models out performed over Encoder Decoder with Attention.



| | Models | BLEU scores |
|---|---|---|
| 0 | Encoder Decoder with Attention | 0.769161 |
| 1 | facebook/mbart-large-50-one-to-many-mmt | 0.921866 |
| 2 | Helsinki-NLP/opus-mt-en-dra | 0.921866 |
| 3 | Helsinki-NLP/opus-mt-en-mul | 0.779691 |

**Figure 4.2.5: BLEU scores for all four models**

## 5 Analysis

In order to better understand our models in terms of learning, the ability to handle long sentences, and attentional architecture choices, we conduct extensive analysis. All of the results presented here are in English-Telugu.

### 5.1 Effects of Translating Long Sentences

We follow (Bahdanau et al., 2015) to group sentences of similar lengths together and compute a BLEU score per group. This demonstrates that our attentional models are more effective in handling long sentences than non-attentional models: the quality does not deteriorate as sentences become longer. In all length buckets, our best model (the blue + curve) outperforms all other systems.

## 5.2 Choices of Attentional Architectures

We explore a variety of models, including an encoder decoder model with attention and different pre-defined hugging face transformer models. We are unable to test all possible combinations due to a lack of resources.

## 5.3 Sample Translations

We compare our NMT systems in the English- Telugu task with various other predefined models. We only show the outcomes of a few of our attention models. We achieve progressive improvements when
(a) using the "Encoder-Decoder model with attention" model has +0.85 BLEU and making our model slightly better than the base attentional system of Bahdanau et al. (2015) (row RNNSearch).
(b) using "facebook/mbart-large-50-one-to-many-mmt", +0.85 BLEU.
(c) using "Helsinki-NLP/opus-mt-en-dra"+0.77 BLEU.
(d) using "Helsinki-NLP/opus-mt-en-mul" +0.77 BLEU.

## 6 Conclusion

In this paper, we propose different mechanisms for Neural Machine Translation: the global approach that looks at all source positions at all times, and a predefined model. When compared to the Helsinki-NLP/opus-mt-en-dra and facebook/mbart-large-50-one-to-many-mmt models, Helsinki-NLP/opus-mt-en-mul, predefined models outperformed this task. We put our models to the test in NMT translation tasks between English to Telugu.

## 7 Future Developments

- OpenAPI GPT is the most powerful Transformer to perform language translation. Because of the potential for misuse, the developers have refused to open source the entire model.
- Starting from the very bottom of a deep neural network, BERT represents "bank" using both its previous and next contexts — "I accessed the... account" — making it deeply bidirectional. BERT will also be the most trending technique for Machine Translation.

## References

[Luong et al.2015] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*.

[Sutskever et al.2014] I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

[KalchbrennerandBlunsom2013] N.Kalchbrennerand P. Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

[Jean et al.2015] Se´bastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural ma- chine translation. In *ACL*.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Ben- gio. 2015. Show, attend and tell: Neural image cap- tion generation with visual attention. In *ICML*.