Attention Transformer Model for Translation of Similar Languages

Farhan
National University of Computer and Emerging Sciences, Karachi Campus
k180900@nu.edu.pk

Muhammad Rafi
National University of Computer and Emerging Sciences, Karachi Campus
muhammad.rafi@nu.edu.pk

Abstract

This paper illustrates our approach to the shared task on similar language translation in the fifth conference on machine translation (WMT-20). Our motivation comes from the latest state of the art neural machine translation in which Transformers and Recurrent Attention models are effectively used. A typical sequence-sequence architecture consists of an encoder and a decoder Recurrent Neural Network (RNN). The encoder recursively processes a source sequence and reduces it into a fixed-length vector (context), and the decoder generates a target sequence, token by token, conditioned on the same context. In contrast, the advantage of transformers is to reduce the training time by offering a higher degree of parallelism at the cost of freedom for sequential order. With the introduction of Recurrent Attention, it allows the decoder to focus effectively on order of the source sequence at different decoding steps. In our approach, we have combined the recurrence based layered encoder-decoder model with the Transformer model. Our Attention Transformer model enjoys the benefits of both Recurrent Attention and Transformer to quickly learn the most probable sequence for decoding in the target language. The architecture is especially suited for similar languages (languages coming from the same family). We have submitted our system for both Indo-Aryan Language forward (Hindi to Marathi) and reverse (Marathi to Hindi) pair. Our system trains on the parallel corpus of the training dataset provided by the organizers and achieved an average BLEU point of 3.68 with 97.64 TER score for the Hindi-Marathi, along with 9.02 BLEU point and 88.6 TER score for Marathi-Hindi testing set.

1 Introduction

This paper focuses on establishing a neural machine translation model using Encoder-Decoder architecture to translate between Hindi-Marathi sentence pairs. We are utilizing an attention mechanism approach by employing the combination of recurrence based RNN Encoder-Decoder layers (Cho and et al., 2014) and latest machine translation technique the Transformers (Vaswani et al., 2017) to address the complexity of WMT-20 similar language data-set (Hindi-Marathi). The motivation behind combining RNN and Transformer architecture is taken from the paper (Huang et al., 2020) to utilize the benefits of both. The formulated machine translation system has numerous applications. It can be used in advertising campaigns to reach native users. The media industry can employ this technology for generating subtitles and broadcasting multilingual news to cover a wide range of native subscribers of a region. Moreover, these systems can provide native vernaculars in social media to increase the activities of indigenous individuals of a native language. Additionally, Search engines can also adopt this method to display relevant results to clients in their region-specific native language.

The paper is composed of three further sections. The second section will display our proposed approach to solve the problem of Hindi-Marathi translation, followed by a discussion on designed experiments for training the model and mechanism for generating the results. Lastly, we will illustrate contributions and future work.

2 Methodology

The paper proposes a novel approach called Attention Transformer shown in Fig. 1 to translate between Hindi and Marathi text. In the figure, two encoder and decoder layers are illustrated. The initial N Encoder-Decoder layers contain RNN units in the form of LSTMs, which get stacked on top of each other. First, the initial encoder layer processes the tokenized input and generates a context vector. The initial decoder layer consumes the context vector generated by the neighboring encoder layer. Plus, it takes the target output as its input while training by utilizing the concept of teacher force training (Goyal and et al., 2016). Next, the processed outputs of both the initial Encoder-Decoder layer gets presented as input to the transformer model.

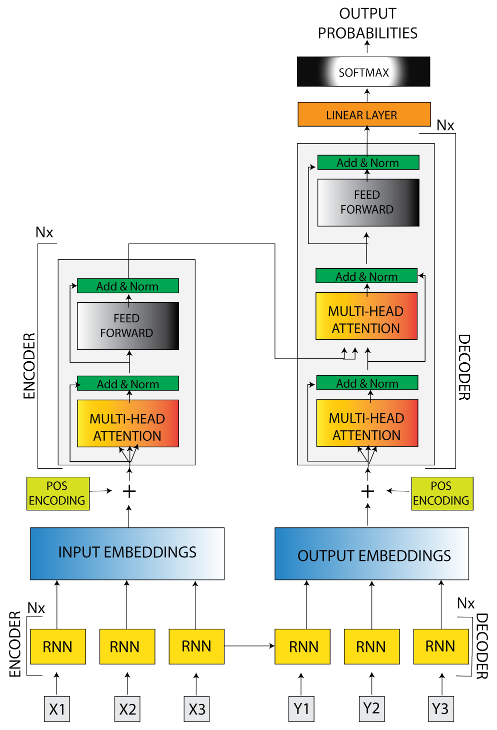

Figure 1: This Figure shows bird's-eye view of the proposed Attention Transformer Model

In the Transformer model, we can observe the second Encoder and Decoder layer stacked on the top of positional input-output embedding layers. In the Transformer, the positional input-output embedding layer receives the output of the initial Encoder-Decoder layer as its input. The role of the positional embedding layer is first to convert its input, which comes from the previous RNN based Encoder-Decoder layer into d-dimensional space where d is the output size of the embedding layer and add a positional encoding vector to it. As a result, all similar words relative to their positions in the training sentences will get cluster together. The working of the positional encoding vector is presented in the paper (Vaswani et al., 2017). Next, the outputs of the positional embedding layer get passed to Encoder-Decoder layers of Transformer. An individual Encoder-Decoder layer of Transformer contains a multi-head attention mechanism followed by a feed-forward layer, and a Transformer can have N number of such layers. The multi-head attention mechanism and feed-forward layer works, as illustrated in the paper (Vaswani et al., 2017), and their outputs are normalized. To train the model, we can apply the teacher forcing mechanism while during testing, a start token is initially required for the decoder to start the decoding process. After that, we can let the decoder to generate the output tokens in a loop by utilizing its current output as input to the next time frame until it produces the end token.

3 Experimental Studies

We have used Two human-level translation evaluation criteria, which are BLEU (Papineni and et al., 2002) and TER (Snover and et al.) scores and two general evaluation metrics that are Sparse Categorical Accuracy and Mean Loss. This section will first discuss the preparation of training dataset and baseline models, followed by training procedures plus their outcomes. And then, we will move towards explaining the results of the testing procedure. It is important to note that all experiments given below are performed using TPU with 180 TFlops, and 64 GB High Bandwidth Memory (HBM) provided by Google Colab, plus an implementation of the experiment is located in the Colab notebook (implementation).

3.1 *Data-set*

Initially, we have a Hindi-Marathi parallel training corpus of 44,685 sentence pairs. We have applied a simple rule to filter out all sentence pairs having the length higher than 24 words. After using this filtering rule on the data-set, we are left with 35,215 sentence pairs on which we have applied 80%-20% split to extract out training and development data. We have separated 100 records from the dev-set, and treat it as unseen data to perform a comparison with the baseline models. The table 1 shows the division of the data-set.

The reason for the maximum length based filtration of the data-set is with an increase in the maximum length of a sentence in the data-set, the complexity of the model increases, hence the training time increases. Although vocabulary size and hyper-parameters of the model also play a significant role in the training time per Epoch. Plus, we are motivated to keep our model simple as much as possible because the quality of the predicted translation gets affected, and it becomes difficult to

DATA-SET CONTENTS	
Data-set total sentence pairs	44,685
Filtered data-set sentence pairs	35,215
Training set sentence pairs	28,172
Development set sentence pairs	6,943
Records for comparison with baseline models	100
Hindi vocabulary size in filtered data	31,417
Marathi vocabulary size in filtered data	53,639

Table 1: The table shows the division of the data-set

debug the model as it grows more complex.

3.2 *Baseline Models*

We have selected the Bahdanau (Bahdanau and et al., 2014) and Transformer (Vaswani et al., 2017) model as a baseline model to compare the performance of our model. We have used their TensorFlow implementation officially given at (TensorFlow, a) and (TensorFlow, b). In our experiment, we have extracted 100 records from the dev-data, which serves as unseen data and helps us to compare the goodness of our proposed model with the selected baseline models. We have trained the model on the Marathi-Hindi dataset, with the mentioned parameters in TensorFlow documentation, and recorded that the Bahadanau model gets an average BLEU score of 0.13, while the transformer gets an average BLEU score of 20.

3.3 *Selecting Hyper-parameters*

The first essential hyper-parameter is to decide the maximum number of words a source or target sentence can have in a single given instance of a training sentence. However, it's a fantasy to develop a model that handles infinite words in the training instance. But as a result, it leads to infinite training time, which is undesirable. We have run the attention transformer model with the top 1000 records after filtering the dev-set with the various maximum number of words a source and target sentence can have and recorded their training time as shown in the chart below. In Fig 2, we can notice that increment in the maximum number of words a sentence can have produces a drastic impact on training time. We have selected 24 as the maximum number of words a sentence can have in our data-set to achieve comparable performance in practical training time.

After filtering the data-set based on the maximum number of words, a sentence can have. The next essential thing is to choose an appropriate batch size for the model. We have executed the Attention transformer model with different batch sizes on dev-set and noted their impact on training time per epoch. The Fig. 3 illustrates that the increment in batch size helps to reduce training time up to an extent after that it reduces the efficiency of the model. We have selected 16 as batch size, as it gives minimum training time per epoch for the model.

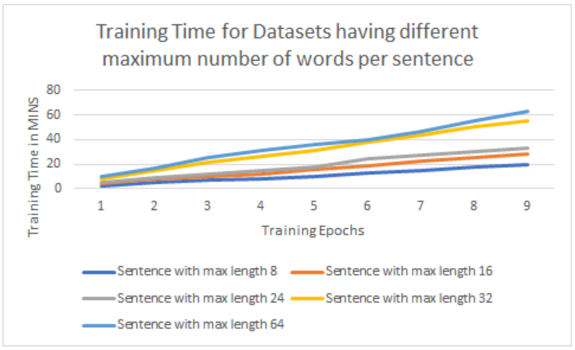

Figure 2: The figure shows line graphs illustrating the training time on filtered data-sets having different lengths of maximum words in a sentence.

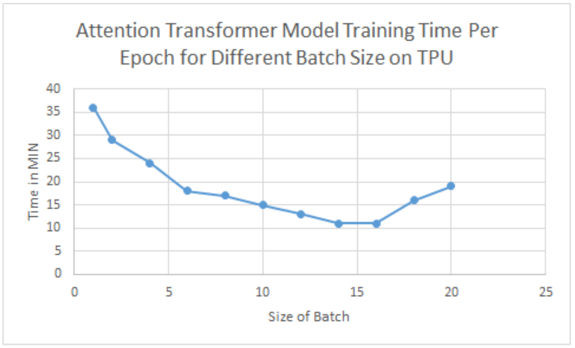

Figure 3: The figure shows a line graph illustrating the training time per epoch for different batch sizes.

In addition to that, to keep the training time of Attention Transformer practical, with a TPU of 180 TFlops and 64 GB High Bandwidth Memory (HBM). We have set the number of initial RNN based Encoder-Decoder layers to one and the number of the second Transformer Encoder-Decoder layer to two. Plus, the number of attention heads in the multi-head attention layer of the Transformer encoder-decoder layer is set to 8, and all other hyper-parameters for transformer model are kept as suggested in the paper (Vaswani et al., 2017).

3.4 *Training the Model*

In training, we have not augmented the original form of the given sentences in the data-set. In the pre-processing step, we have only removed punctuation from the sentences and fed the filtered data to the model, which sums up to 35,215 sentence pairs. We have used the selected hyper-parameters from previous subsections to train the model, which are obtained by optimizing dev data. In addition to that, to keep track of our models' performance, we have used the Sparse Categorical Cross Entropy function as our loss function for evaluating training predictions, which is an integer version of the Categorical Cross Entropy function the details can be observed in the notebook (implementation)

3.5 *Dealing with Over-Fitting and Unseen Vocabulary at Test Time*

To save the model from overfitting, we have kept the training procedure straight-forward by applying a simple rule to train the model until it provides a BLEU score of 0.7 or the performance of BLEU score asymptotes after ten epochs. While training, we have collected average BLEU scores, TER score, Accuracy, and Mean Loss across batches over an epoch to track the performance of the model as shown in (implementation).

Moreover, to deal with new input vocabulary at test time, we have employed a simple trick by generating a miscellaneous token at the time of tokenization. The miscellaneous token gets included in the vocabulary of the model at train time. And the model learns to deal with this token based on its neighbors. During test time, while tokenization, if the input sentence contains any unseen vocabulary, then we exchange that word with the miscellaneous token.

3.6 *Comparison With Baseline Model*

We have trained the baseline models and our proposed attention transformer model on the Marathi-Hindi data-set at the end of each epoch, we have recorded the training time. This per epoch training time will allow us to measure the quickness in the model to finish a training epoch. Fig. 4 below states the comparison of cumulative training time of all three models. It can be seen clearly that the Bahdanau model takes a huge amount of training time as compared to the other two models. We were able to run only 10 epochs for the Bahdanau model in approximately 9 hours. On the other hand Transformer and Attention, Transformer models are very quick, it takes approximately 7 minutes to train an epoch of both the models. However, over the time it can be seen in the graph that the baseline Transformer model is slightly quicker than our proposed Attention Transformer model.

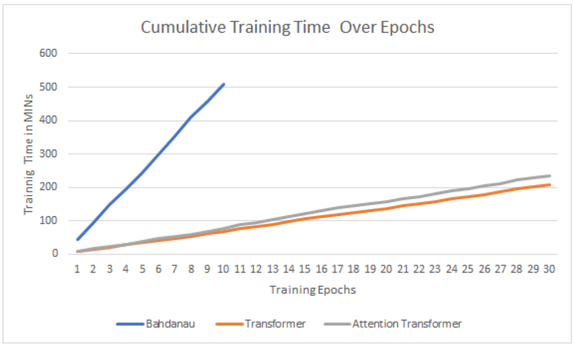

Figure 4: The figure shows comparison of cumulative training time of Bahdanau, Transformer and Attention Transformer model.

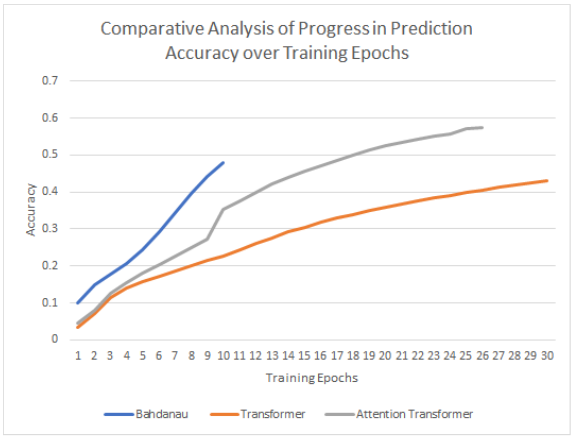

Figure 5: The figure shows comparison of progress in Sparse Categorical Accuracy as we continue to train the Bahdanau, Transformer and Attention Transformer model.

Next, to measure the progress in translation performance as we continue to train the model, we have recorded average Sparse Categorical Accuracy, BLEU score, Sparse Categorical cross entropy loss, and TER scores at the end of each epoch as shown in Fig. (5, 6, 7, 8) respectively. This track of per epoch training performance helps us to visualize the progress of the model in learning the translation probability distribution, plus we can also utilize this information to find out the most active model that fits translation distribution in the least number of epochs.

In the Fig. 5 and 6, we can notice that the Bahdanau model is quickest to adapt translation probability distribution compared to other models as it has shown approximately exponential increment in accuracy and BLEU scores over the initial training epochs. The transformer model is following a relatively linear path in learning the probability distribution.

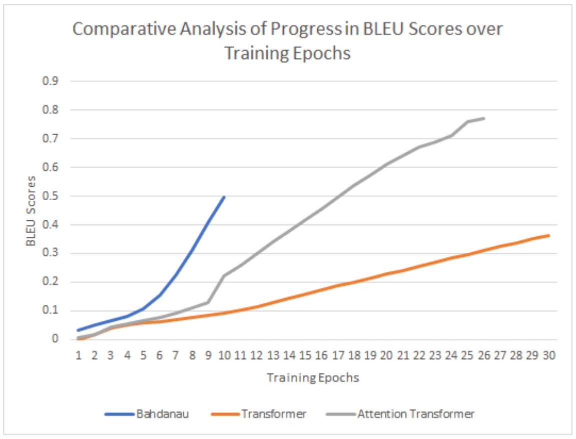

Figure 6: The figure shows comparison of progress in BLEU score as we continue to train the Bahdanau, Transformer and Attention Transformer model.

The reason behind the wining performance of the Bahdanau model is it's inherent recursive nature to model the sequence to sequence tasks, which helps it to learn the positional order of the given sequence. The Transformer lacks this recursive nature and uses a sinusoidal positional encoding scheme to get the awareness of the position of a word in a sentence, which is not as effective as Bahdanau's inherent recursive nature. But this recursive nature hinders the Bahdanau model to exploit parallelism due to this Bahdanau model takes more time to finish a training epoch as compare to the Transformer model.

The Attention Transformer takes the benefits of both the Bahdanau and the Transformer model. The initial layer of RNN helps the Attention Transformer to learn the positional order of the given sequence, plus the stacked Transformer above it allows the Attention Transformer to apply maximum parallelism. In the Fig. 5 and 6, we can notice that the Attention Transformer has given a relatively intermediary performance as compare to the other two models because we have kept the number of RNN and Transformer layers almost equal. If we increase the number of RNN layers in the Attention Transformer model it will start behaving more like the Bahdanau model likewise, if we increase the number of transformer layers then it will act more like the Transformer model.

Similarly, we can use the same argument to reason about the displayed behavior of the performance of the Bahdanau, Transformer, and Attention Transformer model in Fig. 7 and 8

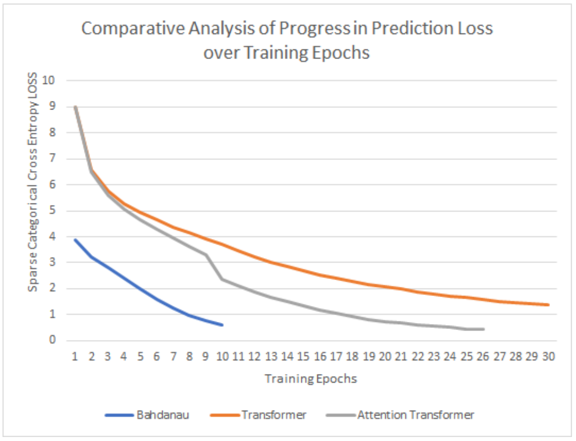

Figure 7: The figure shows comparison of decrements in loss as we continue to train the Bahdanau, Transformer and Attention Transformer model.

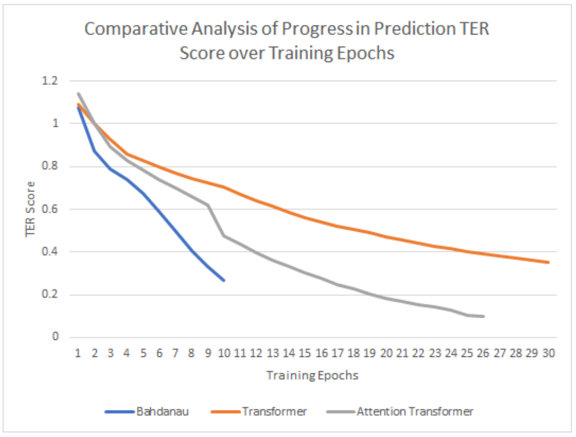

Figure 8: The figure shows comparison of decrements in TER score as we continue to train the Bahdanau, Transformer and Attention Transformer model.

Finally, we have utilized the trained Bahdanau, Transformer, and Attention Transformer model to translate 100 unseen records from Marathi to Hindi, which we have initially separated from the dev dataset. The Fig. 9 below displays the performance of the trained models on the scale of 0-1 BLEU points, the Bahdanau model fails to capture the distribution of unseen data, while the Transformer model performs relatively good. Attention Transformer gives comparatively better performance on average as it shows high BLEU scores for many of the instances.

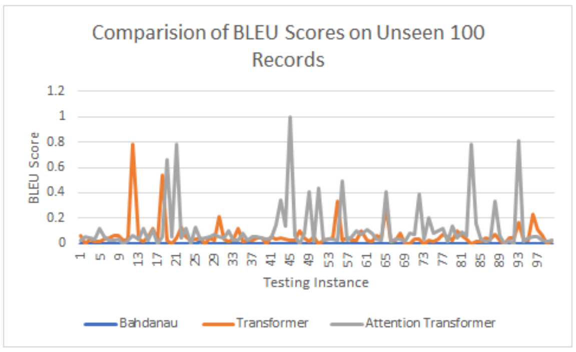

Figure 9: The figure shows comparison of calculated BLEU scores on the scale of 0-1 from the predictions of Bahdanau, Transformer and Attention Transformer model on the unseen 100 records which we have separated from the development data.

3.7 *Testing Results*

We have developed two instance models of Attention Transformer using the procedure mentioned above for both predicting Marathi sentences when Hindi sentences are given as input and predicting Hindi sentences when Marathi sentences are provided as input. We have achieved BLEU points of 3.68 and a TER score of 97.64 for the Hindi-Marathi test pair. Plus, BLEU points of 9.02 and the TER score of 88.68 for the Marathi-Hindi test data-set.

4 Conclusion

This paper has presented a supervised deep neural translation-based approach called Attention Transformer as a tool to perform translation between similar pair of languages (Hindi-Marathi). We have developed a novel Neural Translation method called Attention Transformer to transmute from Hindi source to Marathi and vice-versa by combining the classical recurrence based encoder-decoder approach and Transformers working mechanisms. All supervised translation approaches need parallel corpora as their data-set to learn the probability function of generating translation from source to target. We have solely utilized the WMT-20 Hindi-Marathi parallel corpus as the training data-set for the Attention Transformer model having 44,685 sentence pairs and used two human-level evaluation criteria, BLEU plus TER scores, to evaluate the Attention Transformer model. We have achieved BLEU points of 3.68 and a TER score of 97.64 for the Hindi-Marathi test pair. And, BLEU points of 9.02 with the TER score of 88.68 for the Marathi-Hindi test data-set. The future work under this domain includes applying stochastic optimizations like a genetic algorithm to find the best possible combinations of hyper-parameter to model the probability distribution of source to the target language. Furthermore, we can also stack a reinforcement learning paradigm on a developed supervised neural translation model to create a self-autonomous personalized environment for learning the probability function, which continuously gets updated by taking real-time feedback from the user.

References

Dzmitry Bahdanau and et al. 2014. Neural machine translation by jointly learning to align and translate. Cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Kyunghyun Cho and et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Anirudh Goyal and et al. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4601–4609.

Zhiheng Huang, Peng Xu, and et al. 2020. TRANS-BLSTM: transformer with bidirectional LSTM for language understanding. *CoRR*, abs/2003.07000.

TEAM implementation. Attention transformer hindi-marathi machine translation.

Kishore Papineni and Roukos et al. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew Snover and Bonnie Dorr et al. A study of translation error rate with targeted human annotation. In *In Proceedings of the Association for Machine Transaltion in the Americas (AMTA 2006*.

TensorFlow. a. Tensorflow implementation for bahdanau model online page, accessed: 14.08.2020.

TensorFlow. b. Tensorflow implementation for transformer model online page, accessed: 14.08.2020.

Ashish Vaswani, Shazeer, and et al. 2017. Attention is all you need. In I. Guyon and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.