JUNLP@ICON2020: Low Resourced Machine Translation for Indic Languages

Sainik Kumar Mahata, Dipankar Das, Sivaji Bandyopadhyay
Computer Science and Engineering
Jadavpur University
sainik.mahata@gmail.com, dipankar.dipnil2005@gmail.com
sivaji.cse.ju@gmail.com

Abstract

In the current work, we present the description of the systems submitted to a machine translation shared task organized by ICON 2020: 17th International Conference on Natural Language Processing. The systems were developed to show the capability of general domain machine translation when translating into Indic languages, English-Hindi, in our case. The paper shows the training process and quantifies the performance of two state-of-the-art translation systems, viz., Statistical Machine Translation and Neural Machine Translation. While Statistical Machine Translation systems work better in a low-resource setting, Neural Machine Translation systems are able to generate sentences that are fluent in nature. Since both these systems have contrasting advantages, a hybrid system, incorporating both, was also developed to leverage all the strong points. The submitted systems garnered BLEU scores of 8.701943312, 0.6361336198, and 11.78873307 respectively and the scores of the hybrid system helped us to the fourth spot in the competition leaderboard.

1 Introduction

Machine **T**ranslation (MT) is the translation of one natural language to another using software. Generally, training a good translation system requires the availability of a large and good quality parallel corpus. These corpora are easily available for languages that are spoken globally and have a large digital footprint. But finding the same for less-resourced languages, that are not universally recognized and do not have a large digital presence, is a challenge. This leads to the development of translation systems that do not produce quality results. The present work aims to solve a similar issue and focuses on showing the capability of general domain machine translation when translating into Indic languages, English-Hindi, in our case. The literature includes the description and training process of state-of-the-art translation systems and finally quantifies their performance with respect to the data provided as part of a shared task organized by ICON 2020: 17th International Conference on Natural Language Processing[1].

The shared task was divided into two sub-tasks,

- **SubTask 1** : To show sentence level Machine translation capability for on General domain.
- **SubTask 2** : To show sentence level Machine translation capability for on specified domains.

We took part in the first sub-task and proceeded with developing translation systems with the help of the provided English-Hindi parallel corpus.

Using the provided parallel corpus, we developed three systems. The first two systems was based on **S**tatistical **M**achine **T**ranslation (SMT) and **N**eural **M**achine **T**ranslation (NMT). For training the SMT system, Moses Toolkit (Koehn et al., 2007) was used. The NMT system was a character based seq-to-seq model, that was trained using Bi-Directional **L**ong **S**hort-**T**erm **M**emory (LSTM) cells (Hochreiter and Schmidhuber, 1997). The third system was a hybrid system, that works on the principles of **A**utomated **P**ost **E**diting (APE). In this model, a transformer (Vaswani et al., 2017) based NMT model was used to post edit the outputs, generated by an SMT based translation system.

The rest of the paper is organized as follows. Section 2 describes the parallel corpus that was used to train the above-mentioned translation systems. Section 3 contains the description and the training processes of all the developed translation systems. This will be followed by the evaluation results and discussion in Section 4 and 5. Finally,

[1]https://ssmt.iiit.ac.in/machinetranslation.html

concluding remarks and future scopes have been discussed in Section 6.

2 Parallel Corpus

Multiple English-Hindi parallel corpora were provided by the organizers for training the translation systems. Among these, we decided on using the parallel corpus from CVIT-PIB[2] and CVIT-MKB[3]. Another high-quality corpus from TDIL[4]was also used to train our developed systems. The number of parallel sentences in the CVIT-MKB dataset was 5,272, in the CVIT-PIB dataset were 1,95,208, and in the TDIL dataset were 50,000. In total, we were able to arrange for parallel English-Hindi corpora of 2,50,480 sentences. The data was then tokenized to be used for our further experiments. For tokenizing the English data, NLTK[5] (Bird, 2006) was used and for tokenizing the Hindi data, Indic NLP Library[6] (Kunchukuttan, 2020) was used.

3 Machine Translation

After the English-Hindi parallel corpora were compiled, we proceeded to develop our MT systems. As discussed earlier, the first two MT systems were based on SMT and NMT. The third MT system was a hybrid system, using both SMT and NMT, based on the transformer architecture, and worked on the principle of APE. The description of the all the three systems and the training process for the same is given in Sections 3.1, 3.2 and 3.3 respectively.

3.1 Statistical Machine Translation

For designing the model we followed some standard preprocessing steps on 2,50,480 sentence pairs, which are discussed below.

3.1.1 Preprocessing

The following steps were applied to preprocess and clean the data before using it for training our Statistical machine translation model. We used the NLTK toolkit[7] for performing the steps.

- **Tokenization**: Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens. In our case, these tokens were words, punctuation marks, numbers. NLTK supports tokenization of Lithuanian as well as English texts.

- **Truecasing**: This refers to the process of restoring case information to badly-cased or non-cased text (Lita et al., 2003). Truecasing helps in reducing data sparsity.

- **Cleaning**: Long sentences (No. of tokens $>$ 80) were removed.

3.1.2 Moses

Moses is a statistical machine translation system that allows you to automatically train translation models for any language pair when trained with a large collection of translated texts (parallel corpus). Once the model has been trained, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices.

We trained Moses using 2,50,480 sentence pairs provided by the organizers, with English as the source language and Hindi as the target language. For building the Language Model we used KenLM[8] (Heafield, 2011) with 3-grams from the target corpus.

Training the Moses statistical MT system resulted in the generation of the Phrase Model and Translation Model that helps in translating between source-target language pairs. Moses scores the phrase in the phrase table with respect to a given source sentence and produces the best-scored phrases as output.

3.2 Neural Machine Translation

In order to develop the NMT framework, we decided to employ a character-level neural machine translation system.

The **C**haracter **b**ased **NMT** (CNMT) is based on the architecture as described in Lee et al. (2017) and it relies on the sequence-to-sequence (Sutskever et al., 2014) model. We opted for character embedding based NMT for this task because of the benefits it provides over word embedding based NMT. The benefits, as stated in Chung et al. (2016), are

- capability to model morphological variants

- overcomes out-of-vocabulary issue

[2]http://preon.iiit.ac.in/ jerin/resources/datasets/pib_v0.2.tar
[3]http://preon.iiit.ac.in/ jerin/resources/datasets/mkb-v0.tar
[4]https://tdil.meity.gov.in/
[5]https://www.nltk.org/
[6]https://github.com/anoopkunchukuttan/indic_nlp_library
[7]https://www.nltk.org/
[8]https://kheafield.com/code/kenlm/

- do not require segmentation

The seq2seq model takes a sequence $X = x_1, x_2, ..., x_n$ as input and tries to generate the target sequence $Y = y_1, y_2, ..., y_m$ as output, where x_i and y_i are the input and target symbols, respectively. The architecture of seq2seq model comprises of two parts, the encoder and decoder.

In order to build the encoder, we used four bidirectional layers of LSTM cells. The input of the cell was one hot tensor of English sentences (encoding at the character level). The internal states of each cell were preserved and the outputs were discarded. The purpose of this is to preserve the information at the context level. These states were then passed on to the decoder cell as initial states.

For building the decoder, again two layers of LSTM cell were used with hidden states from the encoder as initial states. It was designed to return both sequences and states. The input to the decoder was one hot tensor (embedding at character level) of Hindi sentences while the target data was identical, but with an offset of one time-step ahead. The information for generation is gathered from the initial states passed on by the encoder. Thus, the decoder learns to generate target data [t+1,...] given targets [..., t] conditioned on the input sequence. It essentially predicts the output sequence, one character per time step.

For training the model, batch size was set to 64, number of epochs was set to 100, activation function was softmax, optimizer chosen was nadam and loss function used was sparse categorical cross-entropy. Learning rate was set to 0.001. The overall architecture is shown in Figure 1.

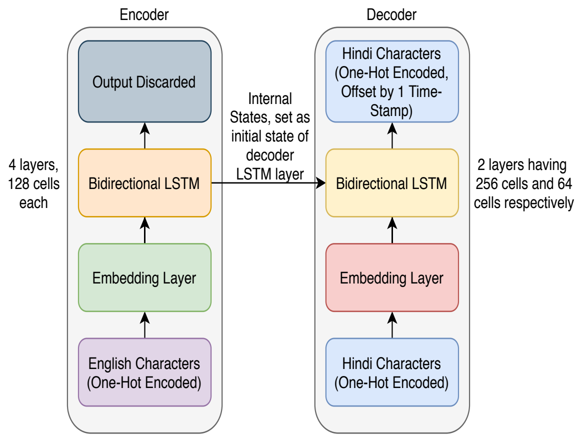
Encoder
Output Discarded
Bidirectional LSTM
Embedding Layer
English Characters (One-Hot Encoded)
4 layers, 128 cells each
Internal States, set as initial state of decoder LSTM layer
Decoder
Hindi Characters (One-Hot Encoded, Offset by 1 Time-Stamp)
Bidirectional LSTM
Embedding Layer
Hindi Characters (One-Hot Encoded)
2 layers having 256 cells and 64 cells respectively

Figure 1: Character based Neural Machine Translation Architecture.

3.3 Hybrid Translation System

The NMT system used for the hybrid translation system is based on the transformer architecture. RNNs typically read one word at a time and perform multiple operations before generating output. But it has been illustrated that the more the number of steps, the harder it is for the network to learn how to make decisions (Bahdanau et al., 2014). Parallelly, RNNs are sequential, and hence taking advantage of parallel computing offered by state-of-the-art computing devices is very difficult.

On the contrary, Transformer models rely heavily on self-attention, thus eliminating the concept of recurrence found in RNN based architectures. In its absence, a positional encoding is added to the input and outputs to mimic the idea of time-steps in a recurrent network. A Transformer model comprises two parts, an encoder, and a decoder, where the encoder is composed of uniform layers, each built of two sublayers; a multi-head self-attention layer, and a position-wise feed-forward network layer. Instead of computing single attention, this stage computes multiple attention blocks over the source, concatenates them, and projects them onto space with the initial dimensionality. On the other side, the feed-forward network sub-layer is a fully connected network used to process the attention sublayers, by applying two linear transformations on each position and a ReLU activation (Vaswani et al., 2017).

The decoder operates similarly, but generates one word at a time, from left to right. The first two steps are similar to the encoder and attend only to past words. The third stage is multi-head attention that attends to these past words, in addition to the final representations generated by the encoder. The fourth stage constitutes another position-wise feed-forward network. Finally, a softmax layer allows the mapping of target word scores into target words. Figure 2 shows the architecture of NMT based on transformer architecture.

For the hybrid model, we intended to merge the SMT and NMT architectures as both these models have their own advantages. So, to incorporate the advantages of both these models into a single system, we decided to merge them in a way that is similar to the APE architecture. For this, we divided the compiled parallel corpus into two parts, one containing 1,50,480 sentences and the other containing 1,00,000 parallel sentences. The first parallel corpus was used to train an SMT sys-

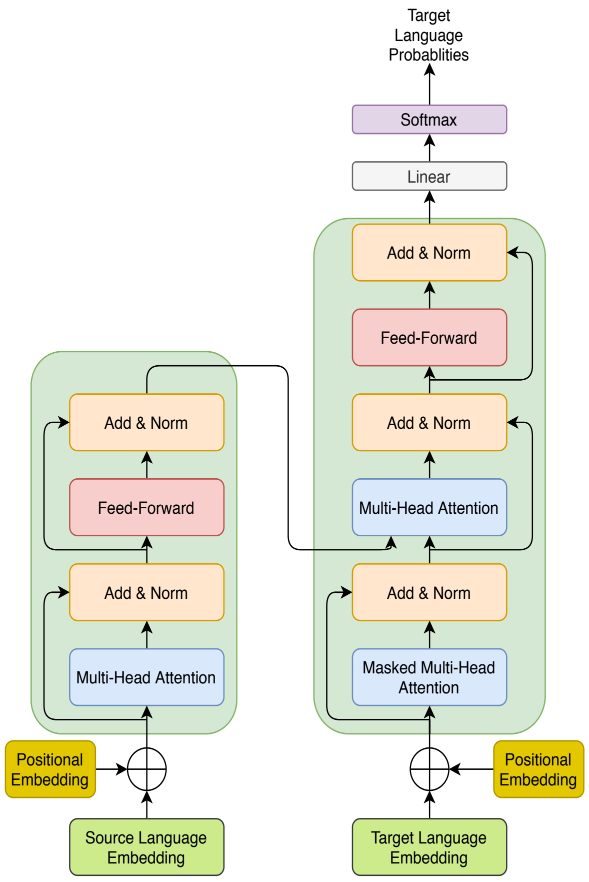

Figure 2: NMT based on Transformer Architecture.

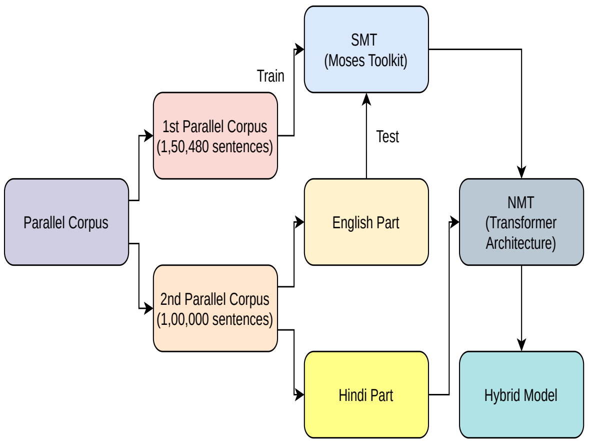
SMT (Moses Toolkit)
Train
Test
1st Parallel Corpus (1,50,480 sentences)
Parallel Corpus
English Part
NMT (Transformer Architecture)
2nd Parallel Corpus (1,00,000 sentences)
Hindi Part
Hybrid Model

Figure 3: Architecture of the Hybrid System.

System	BLEU
SMT	8.701943312
NMT	0.6361336198
Hybrid System	11.78873307

Table 1: Evaluation of the submitted systems.

tem, built using Moses Toolkit. This was done because SMT architectures tend to work well in a low-resource setting. After training the SMT system, the second parallel corpus was used to tune the model. For this, we fed the SMT system with the English part of the second parallel corpus. In turn, the SMT model gave us the translation of these sentences as output. These outputs were then considered as source sentences to an NMT model and the Hindi part of the second parallel corpus was considered as the target. The architecture of the hybrid model is shown in Figure 3.

4 Evaluation

For evaluation purposes, the organizers provided us with a test data of 507 sentences. Upon evaluation, the performance of our systems was calculated using BLEU (Papineni et al., 2002) metric and they are shown in Table 1.

5 Discussion

From Table 1, we can see that SMT performs very well when participating languages belong to a low-resourced setting (Banerjee et al., 2018; Koehn and Knowles, 2017). This is due to the fact that the training data provided by the organizers was small and hence, belonged to similar domains. In general, SMT systems have a higher output quality when trained using domain specific training data since the texts belonging to same domain follow same pattern or usage of words. Also we can see that, during the usage of character based NMT systems, the quality of the output drops drastically. This happens as NMT systems tend to work better when there is a significant overlap between the character set of the participating source and the target languages. Due to the same reason, we see a significant increase in the performance of the hybrid system. This happens, as the second NMT system, that was based on the transformer architecture, is fed with Hindi sentences and learns to map it to Hindi sentences again, during the training process. Hence, there is a significant overlap between the vocabulary sets and hence the increase in performance.

6 Conclusion

The present paper describes the systems submitted to the translation shared task organized by ICON 2020: 17th International Conference on Natural Language Processing. We participated in the English-Hindi translation task and the training data belonged to the general domain. Three systems, SMT, NMT, and a hybrid model was trained using these data. The models were pretty straightforward and did not contain any recent research advancements in the field of Machine Translation. As a future prospect, we would like to experiment with Transfer Learning methods, that learn from large data, and incorporate the knowledge onto models, trained using fewer data. This would be a good option as all the language options of the shared task were Indic languages and good quality and robust multi-lingual translation system can be built out of it.

Acknowledgement

This work is supported by Digital India Corporation, MeitY, Government of India, under the Visvesvaraya PhD for Electronics & IT

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Tamali Banerjee, Anoop Kunchukuttan, and Pushpak Bhattacharya. 2018. Multilingual indian language translation system at wat 2018: Many-to-one phrase-based smt. In *WAT@ PACLIC*.

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, Sydney, Australia. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

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

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. Truecasing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 152–159. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.