Journal of Computer Science

Original Research Paper

Context-Based Machine Translation of English-Hindi using CE-Encoder

Mani Bansal and D. K. Lobiyal

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

Article history
Received: 25-05-2021
Revised: 12-08-2021
Accepted: 28-08-2021

Corresponding Author:
Mani Bansal
School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India
Email: manibansal1991@gmail.com

Abstract: The difficulty in obtaining accurate word alignment and determining a target word that is the best candidate for a source context in machine translation leads to different translations. In this study, we propose a method with a more accurate context model. Our Neural Machine Translation (NMT) approach focuses on the encoder to apprehend the meaning of source sentences for improved translation. The recurrent encoder works by taking into consideration the history and future information of the source context. In this study, we implement the proposed approach into three steps. Firstly, we learn the representation of future context in advance. Secondly, a context-based recurrent encoder called as CE-Encoder with two-level Gated Recurrent Unit (GRU) is used. In this, the bottom-level GRU gathers history data of a sentence and top-level GRU assembles future data information. Finally, the future learned context and the history information from the opposite direction is integrated. The distinguishing factor of the proposed framework from the existing models, specifically Bidirectional Recurrent Neural Network (BiRNN) is that, the current models have not spent substantial time and capacity in learning future context or disambiguating source and target words based on the context which is defined by source sentence. We conduct experiments on the datasets from ILCC and CFILT for the English-Hindi language pair. From the comparative evaluation, we observed that the proposed model outperforms the Bidirectional RNN encoder in terms of translation quality. The proposed model has shown the improvement of 7 Bleu points using the ILCC dataset and 9 points using the CFILT dataset over BiRNN.

Keywords: Neural Machine Translation, Recurrent Neural Network, English-Hindi, CE-Encoder, BLEU

Introduction

Machine Translation (MT) is one of the earliest applications of Natural Language Processing (NLP). Machine Translation makes use of computational linguistics in translating text from one language to another. It involves decoding the meaning of the source text and re-encoding this meaning in the target language. It also helps people from different regions to communicate. Despite being one of the official languages, English is not popular among the Indian population. It is understood only by less than 3% of the Indian people. Hindi, the other official language of the country, is used by more than 400 million people. The different regions in north India, apart from their native language, know only Hindi. Most Indian government records, documents, education, news and historical data are available in English. It is one of the primary reasons that the automatic translation from English to the Indian language is gaining significant importance. However, the English-Hindi translation process raises some challenges due to the nature of languages, such as (i) Indian languages are

morphologically rich. (ii) English is different in terms of word order from the Hindi language. (iii) The availability of parallel corpus for these languages is very limited. Therefore, our research focuses on the machine translation of English-Hindi language pairs. We proposed a context-based Neural Machine Translation (NMT) technique using Context Encoder (CE) for English to Hindi language pairs to address the above challenges. The framework has the best performance in comparison to the baseline methods.

Motivation and Contribution of the Work

The existing research on translation systems was developed using Statistical Machine Translation (Liu *et al*., 2019) based on translation and language models. These models-built translation systems based on phrases or words. The recent advancement in deep neural models has made significant breakthroughs in NLP applications such as language modeling, word embedding and paraphrase detection. NMT (Forcada, 2017) attracted attention and provided promising results on various language pairs. This approach uses an Artificial Neural Network (ANN) to predict the likelihood of a sequence of words while modeling entire sentences in a single integrated model.

Only a few research works Agrawal and Sharma (2017), Parida and Ondřej (2018), Grundkiewicz and Heafield (2018), Saini and Sahula (2018), used the NMT method for English to Hindi translation. Agrawal and Sharma (2017) use a Recurrent Neural Network (RNN) to deal with variable-length input and output by employing Gated Recurrent Units, Long Short Term Memory Units (LSTM), Bidirectional LSTMs and Attention Mechanism. They have not concentrated on the presence of repeated tokens and unknown words (Knowles and Koehn, 2018) presented in the sentence. Parida and Ondřej (2018) target translating short sentences or noun phrases with NMT. They used Bidirectional RNN, shallow and deep sequence-to-sequence and transformer models. They have not explored it on the monolingual and large dataset. Grundkiewicz and Heafield (2018) transliterating named entities that are the phonetic translation of names across languages for which they use deep attention RNN encoder-decoder models. The encoder encodes the input sentence sequence to compute word embedding for vector representation, whereas the decoder decodes that representation into another sequence of symbols. They learned to align and translate simultaneously. Saini and Sahula (2018) also replace LSTM with Bi-LSTM and Deep Bi-LSTM by adding a residual connection. There is a lack of fine-tuning the training of rare and long sentences using smaller datasets in their method.

Our main contribution to the above literature is investigating the pros and cons of employing strategy in translating English to Hindi language pair and highlighting additional opportunities our strategy provides. Considering the above four previous works, all these systems process sentences in isolation and their extended context can prevent mistakes in ambiguous cases and improve translation coherence. The following example illustrates the context-related problem in English sentences:

I Tried to Teach her the Meaning of Fast and Slow

In the above sentence, without knowing the future context of “Slow”, RNN encoding would not know whether the word “Fast” means speedy, vigorous, or abstaining from food. To understand the exact meaning of “Fast”, we must feed the future context information into the encoder. Sometimes the translation of these types of words is done as “व्रत” or as “unk” words. Therefore, in this study, we aim to solve a context-related issue that leads to ambiguity in the Hindi translated sentence and to improve the translation quality. The main contribution of our work can be summarized as follows:

1) Unlike Bidirectional Recurrent Neural Network, where one neural network encodes the forward contextual representation of the sentence and passes this information from history to future, another neural network encodes the backward representation. Our approach gives the future words information to the past words of a sentence, i.e., by using the two combined hidden states, information can be preserved from both the history and the future. We first compute the future context representation in advance and call it CE-Encoder. The CE-Encoder pre-computes semantic knowledge for source words. Then it feeds CE-Encoder in the RNN model with the history of context. Both of them work in opposite directions by using top-level and bottom-level hierarchy
2) In BiRNN, we do not get all the input at the same time. Sometimes, for forward-pass input, backward-pass information is not available. It tends to generate less accurate target sentences. Also, BiRNN passes the stack of layers in forward and backward directions simultaneously. The stacking of layers makes it slower than our model
3) We show that source context is important when translating texts from several domains through the

proposed method and conducted experiments. Our source language, English has Subject-Verb-Object (SVO), whereas target language is Subject-Object-Verb (SOV) form. Therefore, the source-context information is also relevant when the language has source words with the same form that can translate into target words of different form

Related Work

A brief review of the research work carried out in English-Hindi translation is discussed in this section.

Sen *et al*. (2016), authors developed a hierarchical phrase-based Statistical Machine Translation (PBSMT) system for English-Hindi. They performed reordering and augmented bilingual dictionaries to improve the syntactic order of English. PBSMT used an independent reordering model that reorders phrases so that source language word-order and Hindi language word-order would be similar. Banik *et al*. (2020) proposed an alternative SMT technique in which the scores of the phrases from the phrase table are re-balanced by increasing the weight of correct phrases and decreasing the weight of incorrect phrases. The authors Singh *et al*. (2017) presented a translator that uses Translation Memory for English-Hindi. Their translator works on both Fuzzy match and exact match. Translation memory in the translator is a database of segments that are already translated to help human translators. English is divided into segments and matched with the database to fetch the translation from the database. Sharma and Singh (2021) incorporated phrase-based topic model system into baseline phrase-based system. They analysed the effect of topic modelling on general corpus sentences mixed with in-domain text. Jaya and Gupta (2016) considered an Out-Of-Vocabulary (OOV) approach to standardize the dataset. They investigated a corpus augmentation method to improve the translation quality of the bidirectional English-Hindi SMT system. Their strategy worked well for fewer resources without assimilating the external parallel data corpus.

Ambiguities of content and some function words pose challenges to MT systems. These ambiguities forced the authors to explore deep learning (Costa-jussà *et al*., 2017) for the English-Hindi translation system. Narayan *et al*. (2016) presented a quantum neural network machine translator based on the concept of machine learning of semantically correct corpus. The system performed the translation task using its knowledge gained while learning the input sentences from source to target language. The translator acquired the knowledge required for translating in implicit form from the input pair of sentences. Saini and Sahula (2018) investigated the possibility of using shallow RNN (Jang *et al*., 2019) and Long-Short Term Memory (LSTM) based NMT for solving Machine Translation issues. They have used a small dataset and fewer layers for their experiment. They have obtained their results on two-layer and 4-layer LSTM with Stochastic Gradient Descent (SGD). They have compared their results of 2-layer and 4-layer LSTM, stochastic gradient descent with residual connections. Saini and Sahula (2021) also proposed Sequential Adaptive Memory (SAM) model which is an augmented version of the Cortical Learning Algorithm (CLA). They created word pairs, rules and dictionaries for translation but using smaller dataset.

Bhatnagar and Chatterjee (2020) adapted bilingual embeddings and autoencoder networks techniques for English-Hindi language translation.

Ojha *et al*. (2018) performed machine translation for the Indic Languages Multilingual task for the 2018 edition of the WAT shared task. They used English to Hindi, Telugu, Bengali, Tamil languages and a shared Statistical and Neural translation task. Gupta *et al*. (2020) introduced pathological invariance methodology for syntactically similar but semantically different sentences. They replaced one word in a sentence using masked language model and removed word or phrases based on constituency structure. Parida and Ondřej (2018) discussed three NMT models-shallow and deep, sequence-to-sequence and Transformer model. They used a visual genome dataset for translating English to Hindi using out-of-domain datasets of varying sizes. Their target domain was short segments appearing in descriptions of image regions in the visual genome. Kunchukuttan *et al*. (2018) performed training using the neural architecture of transliteration models for multiple language pairs. Each language pair benefits from sharing knowledge with related tasks, i.e., phonetic properties and language writing systems. They used maximal sharing of network components to utilize high task relatedness on account of orthographic similarities: Overlapping phoneme, similar grapheme to phoneme mappings for zero-shot transliteration. Ratnam *et al*. (2021) developed knowledge-based method which is able to handle the linguistic specificities like auxiliary verbs, helping verbs of source and target languages. In Pathak and Pakray (2019) considered optimality in translation through training of the neural network,

using a parallel corpus is having a considerable number of instances in the form of a parallel running English-Tamil, English- Hindi (Agrawal and Sharma, 2017) and English- Punjabi translations. This helps in analyzing the context better (Weissenborn *et al*., 2018) and produce fluency to make NMT a good choice for Indian languages. Very few approaches explored on Context-based English-Hindi Statistical and NMT. Gaikwad (2020) explored the suitability of word to vector (word2vec) and hash to vector (hash2vec) approaches to sequence to sequence text translation using Recurrent Neural Network. The word2vec uses neural network whereas hash2vec is based on hashing algorithm. Gupta *et al*. (2016) presented a methodology for lexical disambiguation in a phrase-based MT system. In that system, source-context is used to extract information from training sentences similar to the sentences to be translated.

Besides this, we have proposed supervised machine learning-based deep learning approach where CE-Encoder is used to solve the Context-based issue compared to the previous research works, which used unsupervised machine learning based approach with deep auto-encoder. To the best of our knowledge, the proposed work is the first to consider this model.

Basic Model- Neural Machine Translation

In this section, we discuss the basics of Neural Machine Translation. The model is attentive to the words of a source sentence, which are more related to the prediction of a target word. It frees a neural network model from compressing source sentences, regardless of their length, into a fixed-length vector. Here, the input sentence is a sequence of words x_1, x_2,....., T_x that needs to be translated and the target sentence is a sequence of words y_1,y_2,……,T_y.

Encoder

The encoder is a bidirectional RNN (Sundermeyer *et al*., 2014) consists of forward and backward RNN's. The forward RNN $\vec{f}$ reads a source sentence $x = x_1, x_2$,……, x_{T_x} from left to right x_1 to x_{T_x} and calculates a sequence of forward hidden states or semantic representation as $\vec{h}_j^T$:

$$\vec{h}_j^T = \vec{h}_1, \vec{h}_2, \vec{h}_3, ..., \vec{h}_{T_x} \quad (1)$$

The backward RNN *f* reads the sequence in the reverse order from T_x x to x_1 and calculates backward semantic representation as $h_{j\text{-}}$:

$$\overleftarrow{h}_j^T = \overleftarrow{h}_1, \overleftarrow{h}_2, \overleftarrow{h}_3, ..., \overleftarrow{h}_{T_x} \quad (2)$$

Then combine forward and backward hidden states $\vec{h}_j^T$ and $\overleftarrow{h}_j^T$ respectively at each time step to build a sequence of annotation vectors $\left(h_1, h_2, ..., h_{T_x}\right)$. Therefore, source annotation h_j contains the summary of both the preceding and following words and focuses on the words around x_j described in Table 1.

$$h_j = \left[\vec{h}_j^T, \overleftarrow{h}_j^T\right]^T \quad (3)$$

Decoder

In the decoder, which is a forward RNN, at each time-step *t*, the soft-alignment mechanism first decides which annotation vectors are most relevant. A relevant weight e_{ij}, for each of the j^{th} annotation vector, are computed with feed-forward neural network *f* that takes previous decoder's hidden state $s_{i\text{-}1}$, j^{th} annotation h_j of input sentence and previous output $y_{i\text{-}1}$:

$$e_{ij} = f\left(s_{i-1}, h_j, y_{i-1}\right) \quad (4)$$

The output e_{ij} are normalized over the sequence of annotation vectors:

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \quad (5)$$

The alignment weight α_{ij} of h_j is the probability that the target word y_i is translated from x_j. The alignment weight α_{ij} of the j^{th} annotation vector is used to obtain context vector c_i of the i^{th} word at target.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (6)$$

The i^{th} context vector c_i is the expected annotation over all the annotations with probabilitie α_{ij}. This way, information can be spread throughout the sequence of annotations, which can be retrieved by decoder accordingly. where:

$s_{i\text{-}1}$ = Decoder's previous hidden state.
$y_{i\text{-}1}$ = Previous output at decoder.
hj = j^{th} annotation of input sentence
α_{ij} = Alignment weight that target word y_i translated from x_j
e_{ij} = Relevance weight
c_i = Context vector of i^{th} word at target

DOI: 10.3844/jcssp.2021.827.847

Figure 1, two Recurrent Neural Networks are represented as encoder and decoder. The Encoder bidirectional RNN consists of forward RNN and backward RNN. The forward RNN reads source sentence "I will visit you in April" ordered as (from x_1, x_2, x_3, x_4, x_5, x_6) and calculates forward hidden states as $\vec{h}_1^{enc}, \vec{h}_2^{enc}, \vec{h}_3^{enc}, \vec{h}_4^{enc}, \vec{h}_5^{enc}, \vec{h}_6^{enc}$. The backward RNN reads input ordered as (from x_6, x_5, x_4, x_3, x_2, x_1) and generates backward hidden states $\overleftarrow{h}_1^{dec}, \overleftarrow{h}_2^{dec}, \overleftarrow{h}_3^{dec}, \overleftarrow{h}_4^{dec}, \overleftarrow{h}_5^{dec}, \overleftarrow{h}_6^{dec}$. Then, annotation h_j as (h_1, h_2, h_3, h_4, h_5, h_6) is calculated for each word by combining forward and backward hidden states. The attention weights of each annotation is calculated as $a_{3,1}$, $a_{3,2}$, $a_{3,3}$, $a_{3,4}$, $a_{3,5}$ and $a_{3,6}$ which is used to obtain context vector c_3 i.e., the target word y_3 translated using input sequence x_1, ….x_6. Context vector c_t of decoder calculates by combining encoder hidden state by using attention weights.

Table 1: Input-Output and Hidden State symbols and their description

Symbols	Description
x_1, x_2, x_3....., x_{T_x}	Input sequence x starting from first symbol x_1 to the last one x_{T_x}
y_1, y_2, y_3……, y_{T_y}	Output sequence y starting from first symbol y_1 to the last one y_{Ty}
$\vec{h}_1, \vec{h}_2, \vec{h}_3$,....., $\vec{h}_{T_x}$	Sequence of forward hidden states of symbol x_1 to x_{T_x}
$\vec{h}_j^T$	Calculates forward semantic representation.
$\overleftarrow{h}_1, \overleftarrow{h}_2, \overleftarrow{h}_3$,....., $\overleftarrow{h}_{T_x}$	Sequence of backward hidden states of symbol x_1 to x_{T_x}
$\overleftarrow{h}_j^T$	Calculates backward semantic representation
h_j	Source annotation encodes the information about j^{th} word with respect to all the other surrounding word in sentence

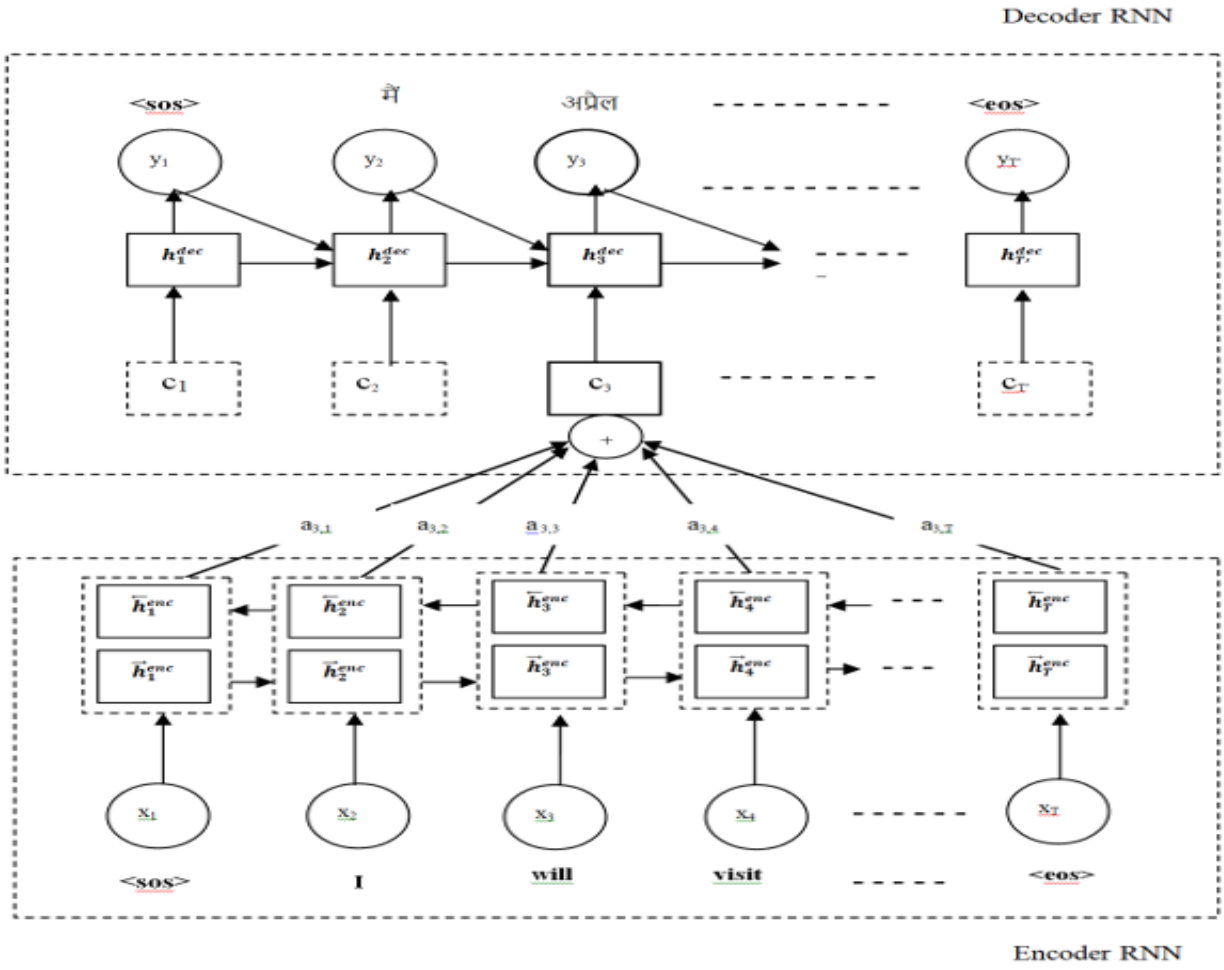


Fig. 1: Neural machine translation with attention

Our Approach

As the hidden representation cannot take future context information sufficiently, it can only encode nearby context. Combining independent forward RNN and backward RNN in Bidirectional RNN directly does not fully use contextual information. Therefore, we discuss the Interdependent Bidirectional Recurrent Neural Network called as CE-Encoder in this section. There are two types of CE-Encoder: FE-Encoder and BE-Encoder. To correctly understand the semantic meaning of a word, we use FE-Encoder if the future context is required. Otherwise, on the other hand, use BE-Encoder to learn the historical context information first. We explain FE-Encoder in detail below, where in the first step, we learn the representation of future context. Then, define context-based recurrent encoder using input token based on learned context representation. Finally, in the third step, we combine these two steps into NMT.

Representation of Future Context

First, the RNN learns the future context of the source sentence. GRU can be considered a simplified version of LSTM. For this, we select the GRU-based recurrent network (Chung *et al*., 2014) in our work due to its capability to remember the processed token. When reading the word "fast", the sentence "I tried to teach her the meaning of fast and slow" has the word "slow" also. The context representation $\overleftarrow{h}_j^e$ induced from right to left given source sentence x at the j^{th} time step and $\overleftarrow{h}_{j+1}^e$ consider to store information of all j+1 inputs. Here, we use e to denote the context.

$$\overleftarrow{h}_j^e = GRU\left(\overleftarrow{h}_{j+1}^e, v\left(x_j\right)\right) \tag{7}$$

Where:

$v(x_j)$ = Source word embedding of x_j

$\overleftarrow{h}_{j+1}^e$ = Initial hidden state while reading the word sequence in backward direction

$\overleftarrow{h}_j^e$ = Hidden state of context representation from right to left

While calculating the update gate (z) for time step j, $v(x_j)$ is multiplied by its own weight W_z. The hidden state $\overleftarrow{h}_{j+1}^e$ holds the information of the future j+1 is multiplied by its own weight U_z. These are added and the sigmoid function is applied. The update gate determines the information that should be passed further. The reset gate (r) in GRU is used to decide how much information to forget. The new memory content uses the reset gate to store relevant information removing the previous time steps information. The final memory content determines what to collect from the current memory content and other previous steps. GRU non-linear transformations are as follows:

[Update gate]: $zj = \sigma(Wz\ v(x_j)) + U_z\ \overleftarrow{h}_{j+1}^e + b_z)$

[Reset gate]: $rj = \sigma(W_r\ v(x_j)) + U_r\ \overleftarrow{h}_{j+1}^e + b_r)$

[Current memory content]: $\overleftarrow{h}_j^e = \tanh\ (W\ v(x_j)) + U[r_j * \overleftarrow{h}_{j+1}^e] + b)$

[Final memory content]: $\overleftarrow{h}_j^e = z_j * \overleftarrow{h}_{j+1}^e + (1- z_j)* \overleftarrow{h}_j^e$

Where:

$\overleftarrow{h}_j^e$ = A candidate activation function for j^{th} token

Context-Based Forward Recurrent Encoder Network

The source sentence representation is obtained by concatenating the forward hidden state of an encoder with future context to produce semantic source representation. For that matter, we use Forward Encoder consists of two-level of hierarchy- Top-level and Bottom-level. The hidden state connected these together to form the final hidden state. It not only looks after long-term dependencies but contains more information on sentences. GRU acts as a bridge to infuse these two kinds of information flow.

1) Bottom level: It summarizes the history information by reading the source sentence. The previous hidden state $\vec{h}_{j-1}^n$ and current input x_j generate internal hidden state $\tilde{h}_j$ through the GRU function.

$$\tilde{h}_j = GRU_{bottom}\left(\vec{h}_{j-1}^n, v\left(x_j\right)\right) \tag{8}$$

Where, n is used to represent CE-Encoder.

2) Top level: It assembles $\tilde{h}_j$ history information given by bottom level and the future context $\overleftarrow{h}_j^e$ computed in above section, together with another GRU function.

$$\vec{h}_j^n = GRU_{top}\left(\tilde{h}_j^n, \overleftarrow{h}_j^e\right) \tag{9}$$

where:

$\vec{h}_{j-1}^n$ = Previous hidden state while reading the word sequence in forward direction

$\vec{h}_j^n$ = Hidden state of context representation from left to right

$\tilde{h}_j$ = Internal hidden state

Integration of FE-Encoder with Neural Machine Translation

We suppose that FE-Encoder encodes history information from left to right, whereas the future context

is encoded from right to left. Whether the history and future context are related to each other should be encoded in the opposite direction. The conventional Bidirectional RNN connects forward RNN and backward RNN in $(j\text{-}1)^{th}$ layer and sent it to j^{th} layer. In contrast, in our method, after getting future context and history, it combines to give the final hidden state for $(j+1)^{th}$ layer. Our encoder uses the independent forward and backward information from the j^{th} layer, making them no longer independent from each other at $(j+1)^{th}$ layer. Therefore, previous layer information is utilized for the accurate content of the current layer.

FE-Encoder

It is called Forward Encoder. It reads the source sentence left to right and the future context in the opposite direction. The future context calculated previously is represented as the Eq. 7 and the summarization of history information by FE-Encoder is represented as the Eq. 8. Substituting both Eq. 7 to 9 gives the forward Encoder as $\vec{h}_j^n$. Figure 2, the GRU reads the sentence "I tried to teach her the meaning of fast and slow" from right to left and generate the future hidden state $\vec{h}_j^e$. The future state saved the long-term future information of a sentence. On the other side, the FE-Encoder reads the same sentence from left to right and generated $\tilde{h}_j$ hidden state. Then both the hidden states are merged to give $\vec{h}_j^n$ exact meaning of the sentence.

$$\vec{h}_j^n = FE\_Enocder\left(\vec{h}_{j-1}^n, v\left(x_j\right), \vec{h}_j^e\right) \tag{10}$$

Backward CE-Encoder

Above, we expressed the required computations for FE-Encoder. Now, the explanation of BE-Encoder is given as follows.

BE-Encoder

It is called Backward Encoder. As compared to Forward Encoder, it operates in the opposite direction of both future context and FE-Encoder. It first learns the historical context, then combines it with future information. The historical context represented in Eq. 11 and future information using another GRU is represented in Eq. 12. By combining Eq. 11 and 12 and inserting them in Eq. 13 gives the Backward Encoder. Because it works in reverse order than FE-Encoder, therefore, in Fig. 3, the GRU reads the sentence from left to right and generate the history hidden state as $\vec{h}_j^e$, whereas, the BE-Encoder reads the same sentence from right to left and generate $\tilde{h}_j$ hidden state. Then both the states are merged to give $\vec{h}_j^n$ hidden state.

$$\vec{h}_j^e = GRU\left(\vec{h}_{j-1}^e, v\left(x_j\right)\right) \tag{11}$$

$$\tilde{h}_j = GRU_{bottom}\left(\tilde{h}_{j+1}^n, v\left(x_j\right)\right) \tag{12}$$

$$\tilde{h}_j^n = GRU_{top}\left(\tilde{h}_j, \vec{h}_j^e\right) \tag{13}$$

$$\tilde{h}_j^n = BE\_Enocder\left(\tilde{h}_{j+1}^n, v\left(x_j\right), \vec{h}_j^e\right) \tag{14}$$

Bidirectional RNN in Fig. 4 generates the forward hidden states as $\vec{h}_j^e$ and backward hidden state as $\tilde{h}_j^n$ by using the same gated recurrent unit for both history and future contextual information. Then, it connects both the hidden states $\left(\vec{h}_j^e, \tilde{h}_j^n\right)$ together to generate the output information. In BiRNN, the output layer gets the information from past and future states simultaneously i.e. it does not save both left-to-right and right-to-left information independently in advance. If any of the information is missing either forward or backward, the translation would not be accurate. Sometimes, we may not get all the inputs all at a time. Let suppose at time $t+2$ we should have input x_1 at time t, x_2 at time $t+1$ and x_4 at time $t+3$ to compute output y_3 at time $t+2$. Therefore, it lags behind to translate long-term dependency sentences, whereas our method computes the final hidden state by analysing the forward and backward information independently.

Pseudocode of our Proposed Approach

1. The encoder network hidden state is generated as: $\vec{h}_j^n = FE\_Enocder\left(\vec{h}_{j-1}^n, v\left(x_j\right), \vec{h}_j^e\right)$ and decoder network has hidden state as: $e_{ij} = f(s_{i-1}, h_j, y_{i-1})$ for the output word at position i

2. Then alignment score are calculated between the decoder previous hidden state and each of the encoder hidden state. The score for each encoder state is represented in a single vector and then softmax: α_{ij} = align (y_i, x_j):

$$\alpha_{ij} = \exp\left(score\left(s_{i-1}, h_j\right)\right) / \sum_{k=1}^{T_x} \exp\left(score\left(s_{i-1}, h_k\right)\right)$$

The alignment model assigns score α_{ij} to input at position j and output at position i.

3. The encoder hidden state and alignment score are multiplied to form a context vector: $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$. The context vector computes the final output of the decoder

DOI: 10.3844/jcssp.2021.827.847

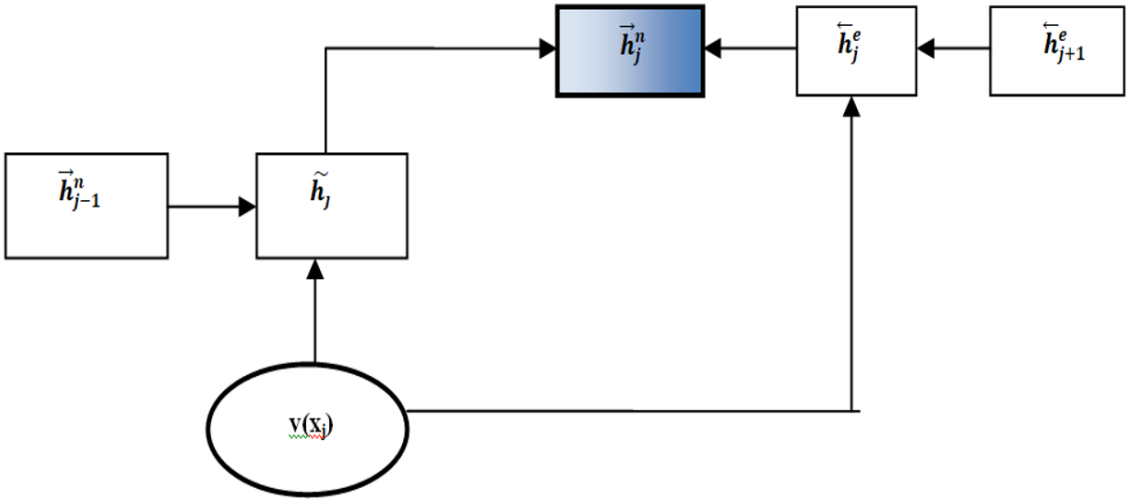


Fig. 2: Forward CE-Encoder

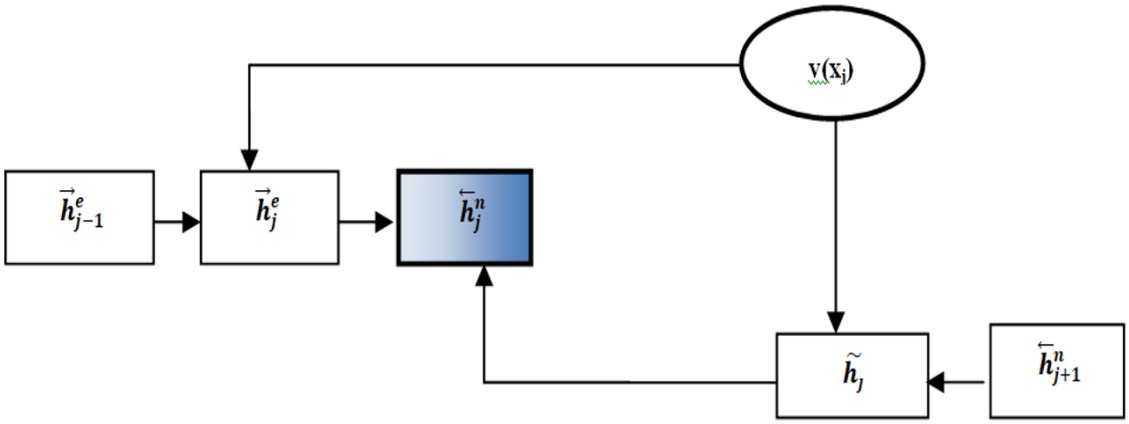


Fig. 3: Backward CE-Encoder

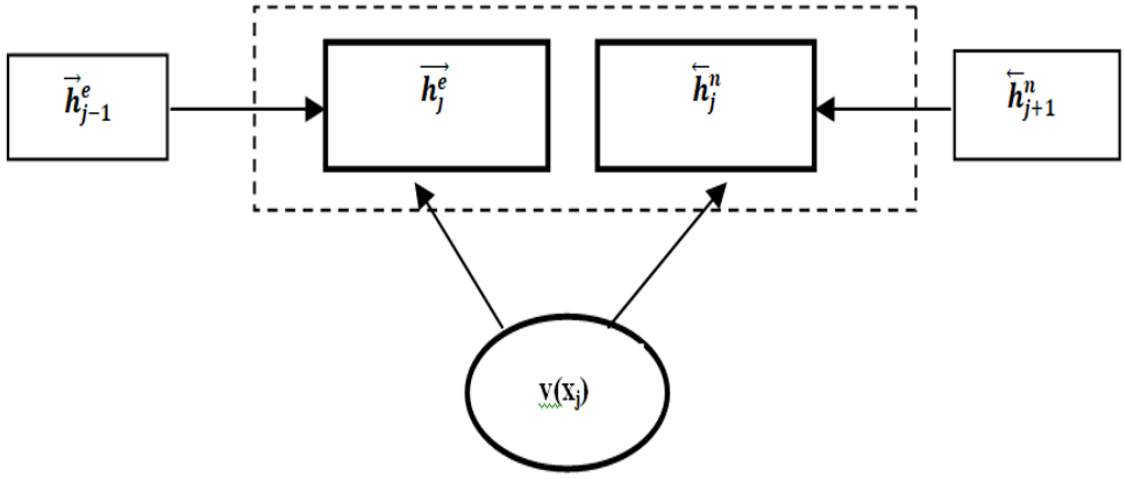


Fig. 4: Bidirectional recurrent neural network

Experiments

In this section, the description of the experimental setup and details of the datasets used for training the translation of English-Hindi language pair using the proposed methodology is given as follows.

Dataset Details

The initial condition for setting up the MT system is the availability of a parallel corpus for the source and target languages. An NMT system has been trained using source-target sentence pairs where English is the source and Hindi is the target language. We have considered the following datasets for our experiment:

- The first data was taken from the Institute for Language, Cognition and Computation (ILCC), the University of Edinburgh (Ins[1])
- Another dataset has been taken from the Centre for Indian Language Technology (CFILT), IIT Bombay (Res[2])

The dataset with the name of Indic parallel corpora from ILCC contains the sentences translated from Wikipedia. The total number of English-Hindi sentence pairs in ILCC is 43,396. We used 35,396 sentences for the training set, 4000 sentences for the development set and 4000 sentences for testing. The CFILT dataset contains data from multiple disciplines. The training set consists of sentences, phrases as well as dictionary entries, their applications and details. The total number of English-Hindi sentences is 1,495,854. We used 1,468,827 sentences for training set, 12,000 sentences for development set and 12,000 sentences for testing.

Dataset Preprocessing

In the above extracted corpus, we observed the presence of noise in both language sentences. To ensure these do not affect training the translation model, we used Moses toolkit for tokenization and cleaning the English data. Hindi data is first a normalized with Indic NLP library 3 followed by tokenization. All the sentences of length greater than 80 from our training corpus were removed and also excluded the sentence pairs containing URLs. To overcome rare words problem in the corpus, we used Byte Pair Encoding (BPE), as proposed by Sennrich *et al*. (2015). It is a data compression technique that replaces the most frequent pair bytes with a single unused byte in a sequence. By merging frequent byte pairs, we combine character/character sequences. BPE helps compound splitting and suffix, prefix separation, which creates new words of the target language. For training, we lowercased all of our training data and used Moses toolkit true caser during testing.

Training Details

We implemented our model using an open-source *dl4mt*[4] system. Our NMT system is trained on GPU based system using Theano computational framework. The dimensional word embedding and hidden states dimension for both source and target languages are d_w = 512 and d_h = 1000, respectively (Bahdanau *et al*., 2014). The encoder and decoder both have 1000 hidden units each. We initialized non-recurrent parameters randomly according to a normal distribution with a standard deviation of 0.01 and zero mean, but the recurrent square matrices are initialized with Singular Value Decomposition (SVD). We applied gradient norm as 5, batch size of 80 to train our model and Ad delta algorithm (Zeiler, 2012) for optimization. Each Stochastic Gradient Descent (SGD) update is computed using a minibatch of 80 sentence pairs for optimization. The learning rate is set to be 0.0005 and is reduced by half at every epoch. Dropout is applied to 0 on the output layer to avoid over-fitting. Both the datasets are trained for 40 epochs. We used a beam-search algorithm during decoding and set beam size to 10.

We train our dataset on two more versions of word embedding with 256 and 620 units. The training is carried out with the embedding dimension of 256 and the batch size of 128 and with embedding dimension of 620 and batch size of 160 by keeping rest of the parameters same. The datasets with 256 embeddings is trained for 30 epochs and datasets with 620 embeddings is trained for 60 epochs. The words that have not appeared in the vocabulary are replaced with a token "UNK".

Experimental Setup

We compared our proposed model with three systems, i.e., Moses, RNN Search[5] and Transformer:

a) **Moses:** It is an open-source phrase-based Statistical Machine Translation System. It consists of two components, training and decoder. The training process in Moses takes parallel data and uses phrases to infer translation. The decoder finds the highest

[1] http://homepages.inf.ed.ac.uk/miles/babel.html.
[2] http://www.cfilt.iitb.ac.in/downloads.htm.
[3]https://anoopkunchukuttan.github.io/indic nlp library
[4]https://github.com/nyu-dl/dl4mt-tutorial/tree/master/session3
[5]In this study, we use RNNSearch and BiRNN interchangeably.

scoring sentence in the target language corresponding to the source sentence (Koehn *et al*., 2007). For Moses[6], we used both datasets ILCC and CFILT. We used 35,396 sentences for the training set, 4000 sentences for the development set and 4000 sentences for testing of ILCC dataset. On the other hand, we used 1,468,827 sentences for the training set, 12,000 sentences for the development set and 12,000 sentences for testing CFILT. We trained a language model (Rahimi *et al*., 2016) of 4-gram on the target data using SRILM[7] (SRI Language modeling toolkit) with modified Kneser-Ney smoothing (Stolcke, 2002). We used GIZA++ [8] toolkit for the word-alignment of training corpus with the " grow-diag-final-and" option. We also exerted the lexical reordering model with type "wbe-msd-bidirectional-fe-allff" i.e., word-based extraction considering monotone, swap and discontinuous orientation. This model is made for both forward and backward sides and it is conditioned on the source and target languages and it treats the scores as individual features. Apart from this, we kept all other parameters to default settings. A configuration file is generated to translate our test sets and compute Bilingual Evaluation Understudy (BLEU), NIST and METEOR scores.

b) RNN Search/BiRNN: It is an attention-based (Bahdanau *et al*., 2014) neural translation system. The encoder and decoder of RNN Search[9] consist of forward and backward recurrent neural networks. We used the same division of ILCC (43,396) and CFILT (1,495,854) dataset into training, development and test set as used for Moses. The encoder and decoder of RNN Search have 1000 hidden units each. A multi-layer network with a single maxout hidden layer is used to compute the conditional probability of each target word. We used default settings for the rest of the parameters to train the model on both datasets. The trained model is saved after each epoch. Finally, our test set is analyzed using the saved models to compute the BLEU scores

c) Transformer: We used Transformer [10] model (Vaswani *et al*., 2017) with six layers in encoder and decoder networks. The same dataset division used for this also. Each encoder block contains a self-attention layer, followed by two fully connected feed-forward layers with a Rectified Linear Unit (ReLU) non-linearity between them. Each decoder block contains self-attention, followed by encoder-decoder attention and two fully connected feed-forward layers with a ReLU between them. We used word embedding and hidden state dimensions as 512 and 2048 feed-forward inner-layer dimensions with dropout = 0.1. We used Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. Further, we used multi-headed attention with eight attention heads, length penalty $\alpha = 0.6$ and beam search with a beam size of 4. The trained model is saved to test the translation results

d) Our Approach: First, we trained our system using the English-Hindi ILCC training dataset and saved the trained model obtained at 30, 40 and 60 epochs each for different word embedding dimensions. Each of the trained models has been tested using the ILCC test set. The predicted results of translations are provided to BLEU for evaluation (Guzmán *et al*., 2017). Furthermore, we also re-trained our NMT system using the CFILT training corpus. Then we performed the training process as carried out earlier. Further, such a setup helps analyze the change in the behavior of the Machine Translation system with an increasing number of new sentences in the corpus

Results and Analysis

We compared the performance of our proposed model with Moses, RNN Search, Transformer and their different variants as given in the tables below.

Using the ILCC test set of 4,000 sentences, FE-Encoder achieves a 28.56 BLEU score, whereas the BE-Encoder achieves a 28.48 score as the highest score while trained the model using a word embedding with 512 dimensions in Table 2. They score 10 BLEU points over Moses, 7 points more than BiRNN and 2 points over Transformer. FE-Encoder NIST and METEOR scores are 5.95 and 0.58, respectively, from Table 3 and 4. The BE-Encoder gives 6.05 NIST scores more than FE-Encoder, whereas METEOR scores are almost identical for all dimensions. They score 2 NIST scores and 2 METEOR points; moreover, Moses, 1.5 NIST and 1.5 METEOR scores over BiRNN, whereas both NIST and METEOR 1 point over Transformer. Therefore, both FE-Encoder and BE-Encoder outperforms all the evaluation metrics.

Using a CFILT test set of 12,000 sentences, FE-Encoder achieves a 43.52 BLEU score, whereas BE-

[6]http://www.statmt.org/moses/
[7]http://www.speech.sri.com/projects/srilm/download.html
[8]http://www.fjoch.com/GIZA++.html
[9]https://github.com/lisa-groundhog/GroundHog
[10]https://github.com/pytorch/fairseq/blob/master/fairseq/models/fairseq_model.py

Encoder achieves a 44.34 score. This dataset also achieves the highest BLEU score while we use the word embedding dimensions of 512. Therefore, medium size dimensions are sufficient to train the translation model.

They score 13 BLEU points over Moses, 9.5 points more than BiRNN and 2 points more than Transformer scores. FE-Encoder scores using NIST and METEOR are 8.35 and 0.75, respectively. The BE-Encoder gives 8.91 NIST scores more than FE-Encoder, whereas METEOR scores are almost the same. They score 2 NIST scores and 2 METEOR points; moreover, Moses, 1 NIST and 2 METEOR scores over BiRNN and 1 point more than Transformer of NIST and METEOR score. It shows our CE-Encoder is better than other models. The difference between the scores of FE-Encoder and BE-Encoder is minimal. This is because the Forward Encoder reads the sentence left to right and the Backward Encoder reads the sentence from right to left, the difference of only reading direction.

In BiRNN, we do not get the inputs all at once concerning the time. Sometimes, for forward pass input, the backward pass information is not available. Therefore, it gives less accuracy in translating source sentences. The other reason that BiRNN is lagging is behind since it does not encode the entire input sentence into a fixed-length vector. Therefore, BLEU, NIST and METEOR scores are comparably less than our model.

The results shown in Fig. 5, BLEU score of sentences with the number of epochs increased our proposed model. The BLEU score is the highest of the medium size trained model. Therefore, it achieves more accuracy for the translation model. The learning rate of the training network increases with the increase in the number of epochs. BLEU of training data has seen normalizing after 36 epochs, that is, epoch coverage. The best-trained model was obtained at epoch 40 and scores did not improve much after 40 epochs. After that, we group our sentences of similar lengths to evaluate the translation model.

We categorize our test set into six separate groups according to their length of source sentences (0-10), [10-20), [20-30), [30-40), [40-50) and all the sentences above the length of 50 has been kept in another set. NMT models perform better on shorter sentences than long sentences. The results are shown in Fig. 6 satisfy the finding, as our CE-Encoder performs better on short sentences than long sentences. Also, CE-Encoder outperforms BiRNN, Moses and Transformer for all the sentence lengths. Moses performs better on the longest sentences because Moses is based on a phrase-based translation process. But the average size of all models is the same that does not affect the improvement of CE-Encoder. The BiRNN passes the stack of layers in a bidirectional way, in forward and backward directions simultaneously. The stacking of layers makes it slower than our model.

In Table 5, we have shown the BLEU score achieved by RNN Search, Transformer and our model with respect to training time of word embedding of 512 dimension units. Table 6 gives the Bleu score using the CFILT dataset. As we can see from both tables, BiRNN needs more time if we want better results and on the other side, Transformer requires approximately 70 h more to complete the training on the dataset. Hence, Transformer has not achieved a 26.09 and 41.77 bleu scores mentioned in Table 2 using both datasets.

In Table 7, we compared our model with previously applied approaches to the English-Hindi language pairs by using BLEU scores. We divide CFILT 1,492,827 sentences into training, validation and test set. The training set contains 130,000 sentences, validation and the test set contains 35,000 sentences each. The sentences are trained for only 11 epochs. After that, the Bleu score is calculated on the test set. During the comparison, our BE-Encoder achieves a 21.63 BLEU score. Moreover, FE-Encoder outperforms all the other models with a 21.84 BLEU score.

We can see from Table 8, 9 and 10 that performed translation on three types of sentences (short, medium, large) where MosesT, RNN SearchT, TransformerT and CE-EncoderT are transliterated sentences. Moses incorrectly conveys the meaning of sentences and sometimes misses some words. RNN Search does not have fluency comparable to our CE-Encoder. ssssA Transformer is near to translate the small and medium sentences correctly but it can only deal with fixed-size text strings. We can see in the medium translated sentence from Table 9, Moses dropped the words "किया जाता है" after the word "विभाजन" and in RNN Search the position of the word "खिलाड़ी में" changes the meaning of source sentence. In a long translated sentence in Table 10, Moses model translates "seeking to break" as "में टूटा ऑटो बीमा" and RNN Search as "टूटना चाह रहा था". Therefore, they fail to recognize the proper meaning of the source sentence. Their system wants to convey a similar meaning, but the sentence structure is somewhat adequate but not fluent.

For the deep analysis of other systems, the attention weight alignments of RNN Search, Transformer and CE-Encoder for short, medium and long sentences are represented in Fig. 7, 8 and 9. The vertical axis translates English sentences into horizontal aligned Hindi language. For the short sentence in Fig. 7, RNN Search aligned all the words correctly except "He" and

"their", whereas Transformer aligned the complete sentence correctly. As the sentence length increased, the performance of RNN Search and Transformer deteriorates. For the medium sized sentence in Fig. 8(a), RNN Search has no alignment for the word "their" and there is weak alignment for the word "are", which modifies the meaning of the input sentence. Whereas, in Fig. 8(b), Transformer MT has no alignment for the word "are categorized". So, RNN Search and Transformer do not perform the translation thoroughly. For the long translated sentence, in Fig. 9(a) RNN Search has not strong alignment for the words "seeking to break" and Transformer in Fig. 9(b) has not aligned "would pay" correctly. The CE-Encoder aligned the word "seeking to break" to "प्रवेश करने का प्रयास" and "would pay" to "भुगतान करेंगे". We observed that our CE-Encoder solved these problems by preserving all the words to maintain adequacy and fluency better than Moses, RNN Search and Transformer systems.

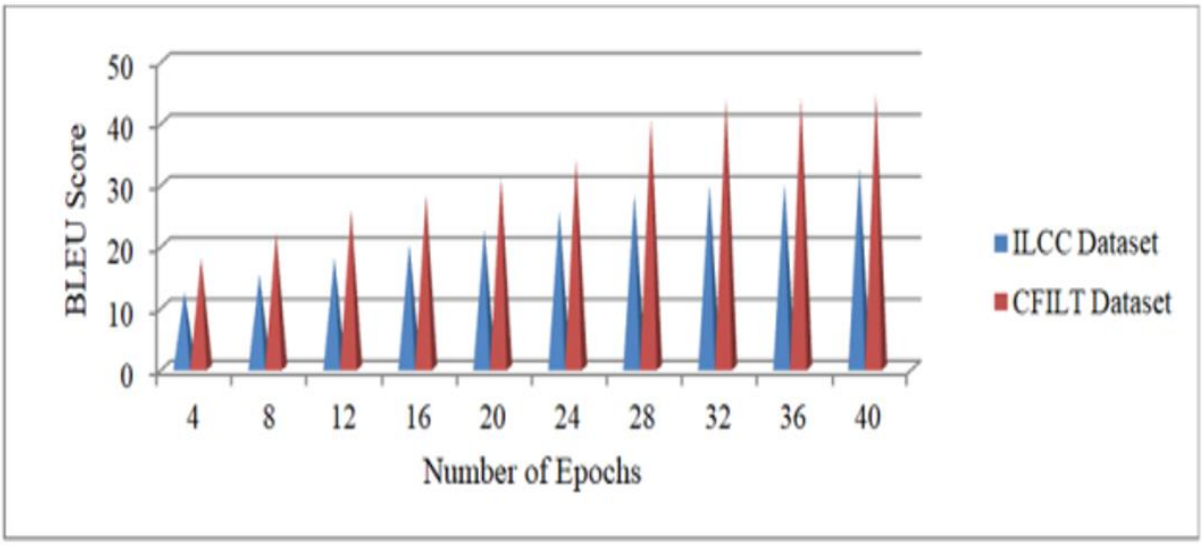


Fig. 5: BLEU score with increasing number of epochs of medium sized trained model

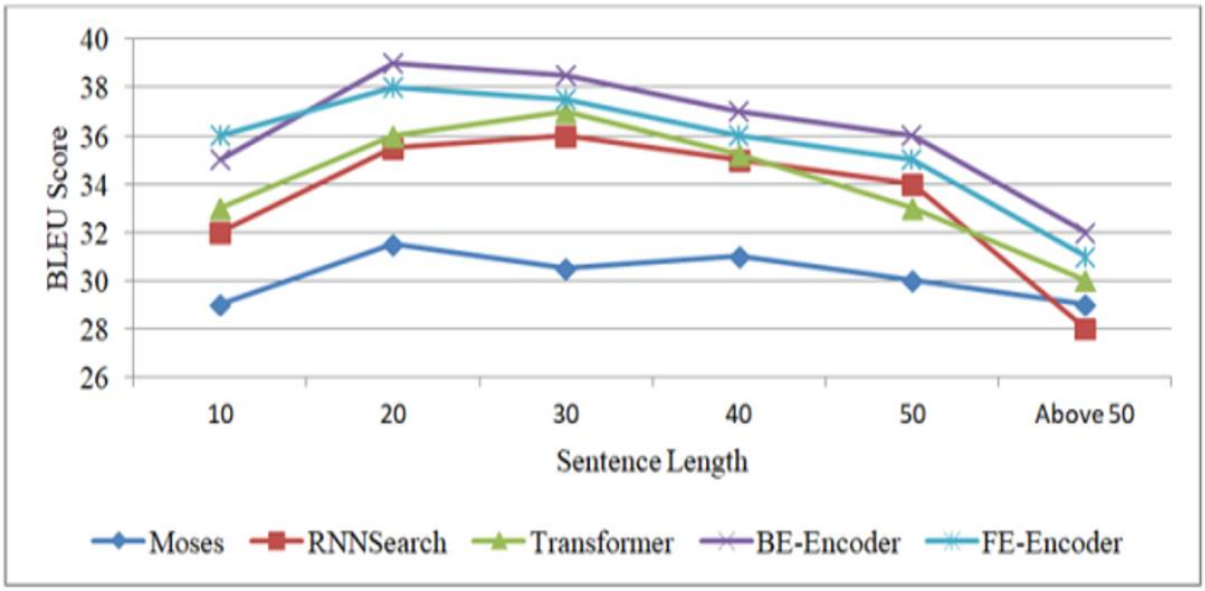


Fig. 6: BLEU score of different length of sentences of various model

DOI: 10.3844/jcssp.2021.827.847

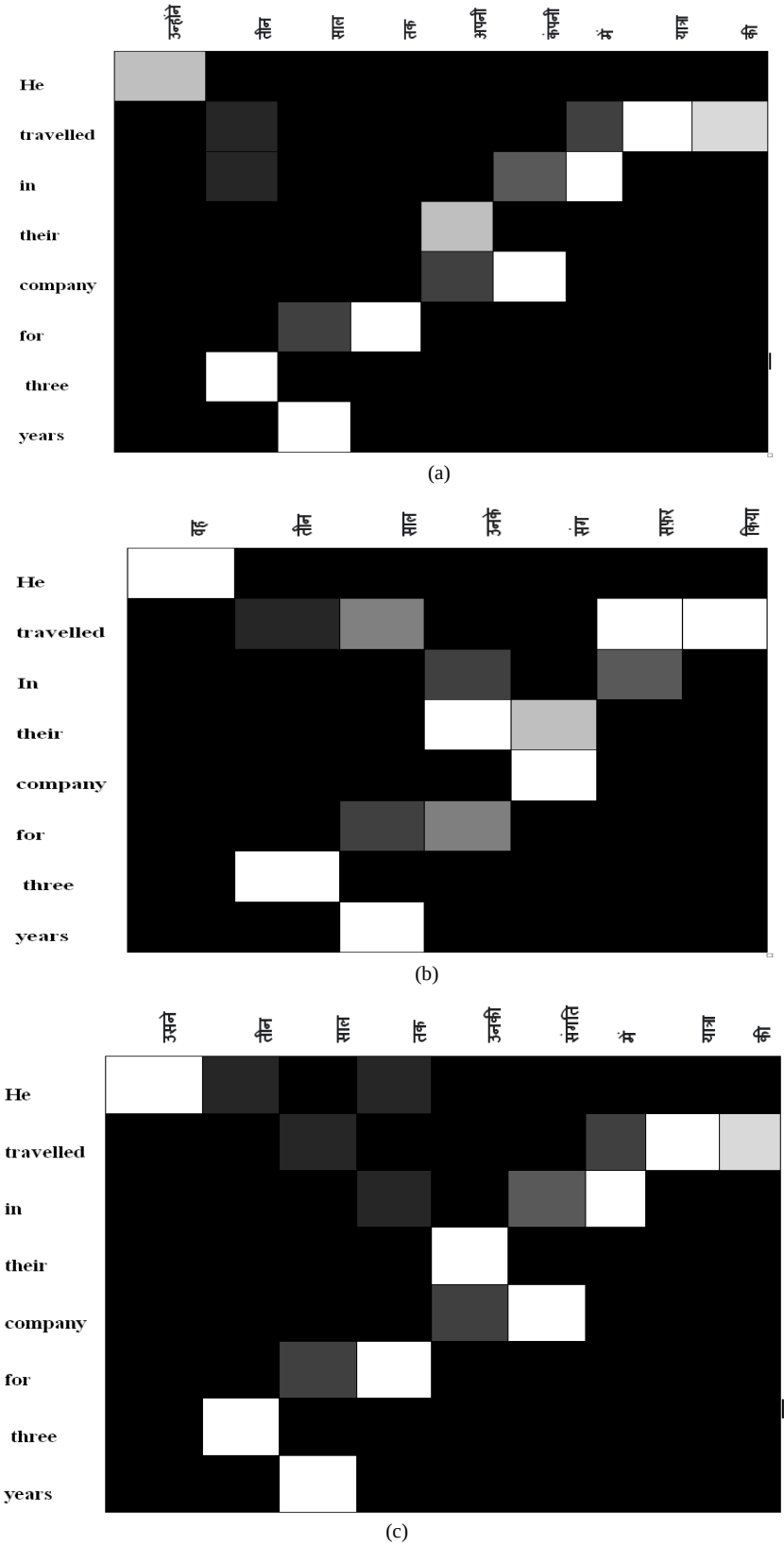


Fig. 7: Alignments found of Short sentence by (a) RNN Search, (b) Transformer and (c) CE-Encoder. The y-axis and x-axis of plot corresponds to the source sentence (English) and generated translation (Hindi), respectively

DOI: 10.3844/jcssp.2021.827.847

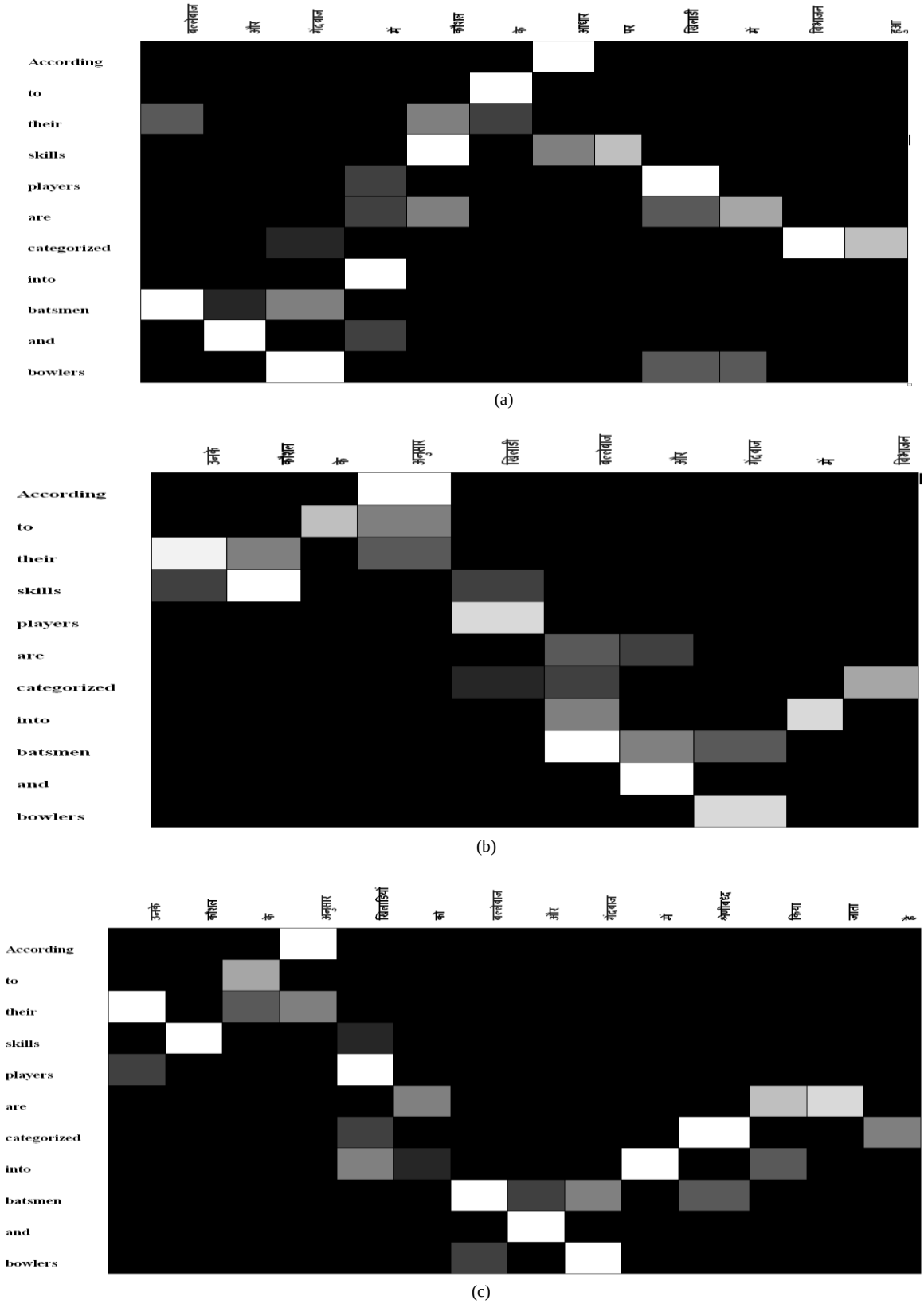
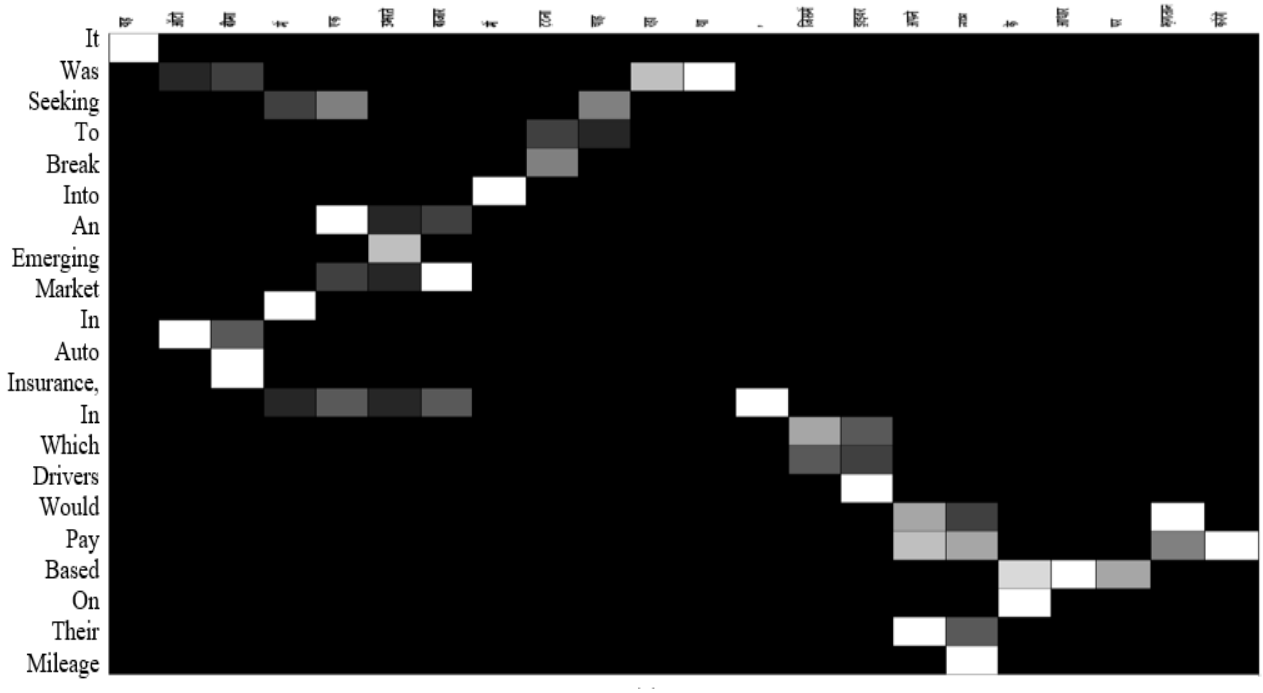
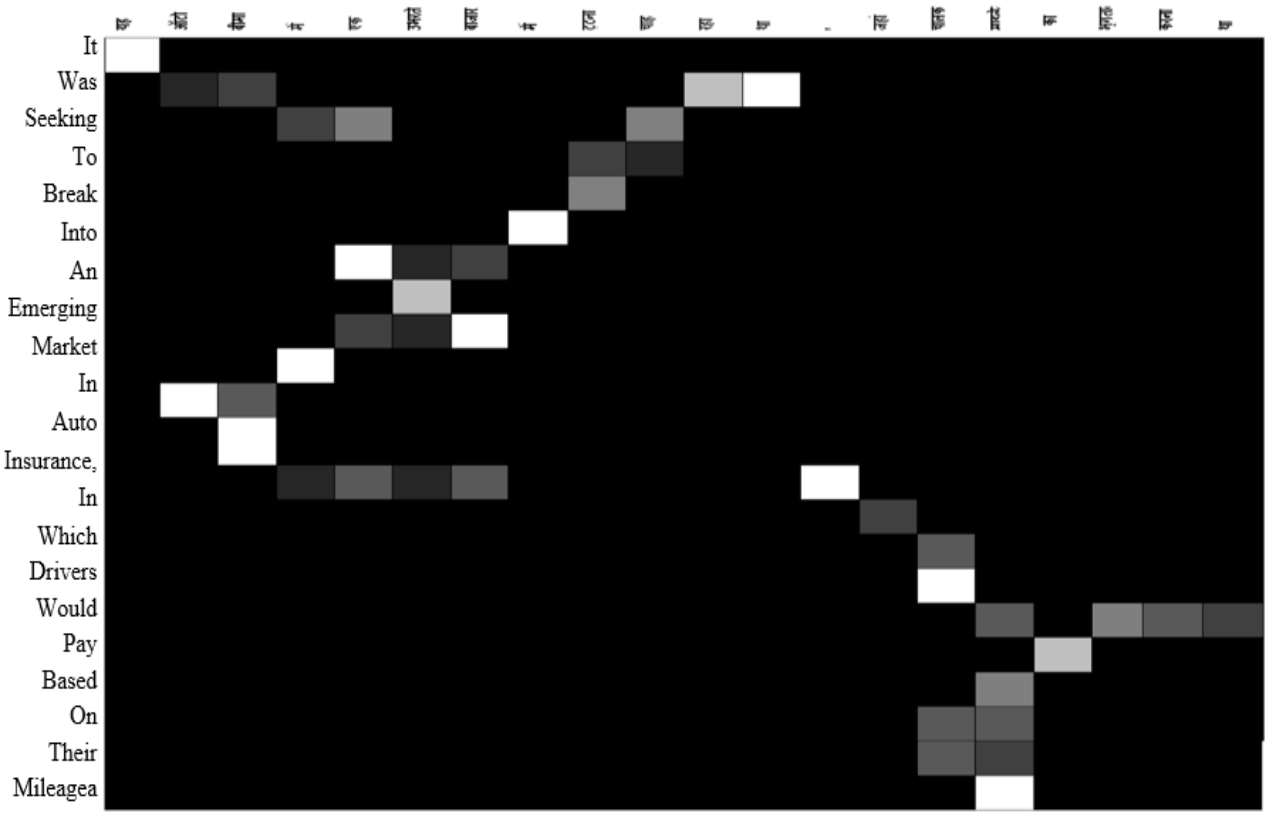


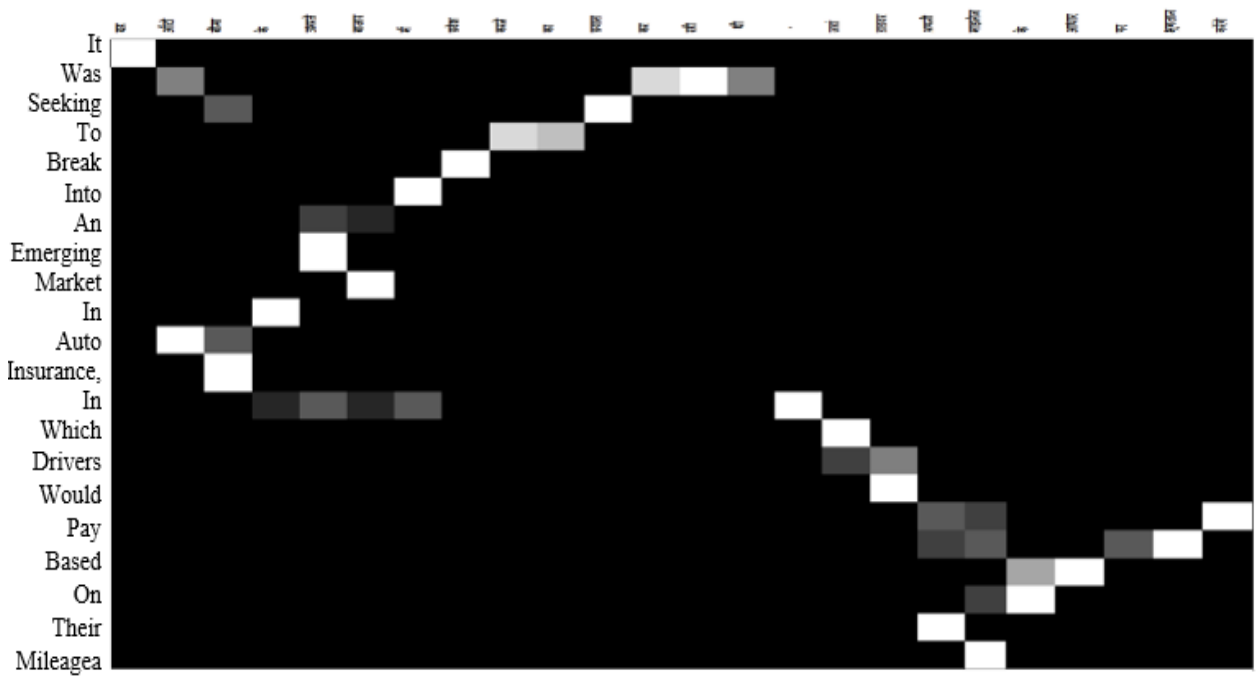
Fig. 8: Alignments found of Medium size sentence by (a) RNN Search, (b) Transformer and (c) CE-Encoder


It
Was
Seeking
To
Break
Into
An
Emerging
Market
In
Auto
Insurance,
In
Which
Drivers
Would
Pay
Based
On
Their
Mileage

(a)


It
Was
Seeking
To
Break
Into
An
Emerging
Market
In
Auto
Insurance,
In
Which
Drivers
Would
Pay
Based
On
Their
Mileagea

(b)



(c)

Fig. 9: Alignments found of long sentence by (a) RNN Search, (b) Transformer and (c) CE-Encoder

Table 2: BLEU score of various MT models

	BLEU Score					
	256 dimensions (small)		512 dimensions (medium)		620 dimensions (large)	
Model	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset
Moses	18.67	30.50	18.67	30.50	18.67	30.50
RNN Search/BiRNN	22.12	33.62	21.11	34.75	19.41	32.37
Transformer	25.88	40.04	26.09	41.77	21.42	35.65
FE-Encoder	**27.31**	**41.17**	**28.56**	43.52	**23.20**	**37.83**
BE-Encoder	27.02	41.01	28.48	**44.34**	22.74	36.96

Table 3: NIST score of various MT models

	NIST Score					
	256 dimensions (small)		512 dimensions (medium)		620 dimensions (large)	
Model	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset
Moses	3.31	6.55	3.31	6.55	3.31	6.55
RNN Search/BiRNN	4.18	7.40	4.28	7.41	3.68	6.83
Transformer	4.82	8.05	5.12	7.97	4.45	7.18
FE-Encoder	**5.95**	**8.35**	5.77	8.62	**5.49**	**7.81**
BE-Encoder	5.63	8.00	**6.05**	**8.91**	4.94	7.70

Table 4: METEOR score of various MT models

	METEOR Score					
	256 dimensions (small)		512 dimensions (medium)		620 dimensions (large)	
Model	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset	ILCC dataset	CFILT dataset
Moses	0.21	0.44	0.21	0.44	0.21	0.44
RNN Search/BiRNN	0.32	0.56	0.28	0.46	0.33	0.50
Transformer	0.41	0.67	0.35	0.57	0.39	0.57
FE-Encoder	**0.48**	**0.75**	**0.44**	0.68	**0.41**	**0.64**
BE-Encoder	0.47	0.72	0.42	**0.71**	0.40	0.62

Table 5: Changes of Bleu score with training time using 512 embedding units of ILCC dataset

Time	RNN Search/BiRNN	Transformer	FE-Encoder	BE-Encoder
1 h 15 min	5.98	8.32	11.87	10.01
10 h 26 min	9.46	12.37	18.54	16.29
21 h 4 min	15.33	17.41	22.61	21.71
29 h 45 min	21.11	22.99	28.56	28.48

Table 6: Changes of Bleu score with training time using 512 embedding units of CFILT dataset

Time	RNN Search/BiRNN	Transformer	FE-Encoder	BE-Encoder
1 h 50 min	10.64	12.23	13.21	14.58
12 h 35 min	18.71	23.17	25.03	27.96
24 h 12 min	23.53	33.20	37.84	39.77
37 h 30 min	34.75	36.76	43.52	44.34

Table 7: Comparison of Bleu score of earlier MT techniques with our model

Architecture	BLEU
2 layer LSTM + SGD	16.75
4 layer LSTM + SGD	17.22
2 layer (Bi-directional) LSTM + SGD	18.30
4 layer (Bi-directional) LSTM + SGD + Res[11]	18.11
Auto Encoder-Latent Semantic Analysis	10.40
2 layer GRU + attention	11.24
4 layer GRU + attention	12.21
LSTM + Neural Semantic Encoder (NSE)	15.68
Non-autoregressive Sequence Generation (Ma *et al*., 2019)	17.64
Phrase and Neural Unsupervised MT (Lample *et al*., 2018)	18.10
Deep Recurrent NMT (Zhou *et al*., 2016)	18.79
NMT in Linear Time (Kalchbrenner *et al*., 2016)	19.40
Evolved Transformer (So *et al*., 2019)	20.55
FE-Encoder	**21.84**
BE-Encoder	21.63

Table 8: Example of English to Hindi translation on short sentence

Source	He travelled in their company for three years
Reference	उसने तीन साल तक उनके संग यात्रा की
Reference$_T$	Usane teen saal tak unkee sang yaatra kee
Moses	उन्होंने तीन साल यात्रा की
Moses$_T$	Unhonne teen saal yaatra kee
RNN Search	उन्होंने तीन साल तक अपनी कंपनी में यात्रा की
RNN Search$_T$	Unhonne teen saal tak apanee kampanee mein yaatra kee

[11]Residual connection

Table 8: Continue

Transformer	वह तीन साल उनके संग सफ़र किया
Transformer$_T$	Veeh teen saal unkee sang safar keeyaa
CE-Encoder	उसने तीन साल तक उनकी संगति में यात्रा की
CE-Encoder$_T$	Usane teen saal tak unkee sangtee mein yaatra kee

Table 9: Example of English to Hindi translation on medium sentence

Source	According to their skills players are categorized into batsmen and bowlers
Reference	प्राथमिक कुशलता के आधार पर खिलाडियों को बल्लेबाज या गेंदबाज के रूप में वर्गीकृत किया जाता है
Reference$_T$	Prathmik kushalta ke adhaar par khilaadiyon ko ballebaaj ya gendbaaj ke rup mein vargikrit kiya jataa hai
Moses	खिलाडी बल्लेबाज या गेंदबाज यक्ति के अनुरूप विभाजन
Moses$_T$	Khilaadi ballebaaj ya gendbaaj yakti ke anurup vibhajan
RNN Search	बल्लेबाज और गेंदबाज में कौशल के आधार पर खिलाडी में विभाजन हुआ
RNN Search$_T$	Ballebaaj aur gendbaaj mein kaushal ke adhaar par khilaadi mein vibaajan huyaa
Transformer	उनके कौशल के अनुसार खिलाडी बल्लेबाज और गेंदबाज में विभाजन
Transformer$_T$	Unke kaushal ke anusaar khilaadi ballebaaj aur gendbaaj mein vibaajan
CE-Encoder	उनके कौशल के अनुसार खिलाडियों को बल्लेबाज और गेंदबाज में श्रेणीबध्द किया जाता है
CE-Encoder$_T$	Unke kaushal ke anusaar khilaadiyon ko ballebaaj aur gendbaaj mein shrenibad kiyaa jataa hai

Table 10: Example of English to Hindi translation on long sentence

Source	It was seeking to break into an emerging market in auto insurance, in which drivers would pay based on their mileage
Reference	यह ऑटो बीमा के अभरते हुए बाजार में प्रवेश करने का प्रयास कर रही थी, जिसमें चालकों को अपनी माइलेज के आधार पर भुगतान करना था
Reference$_T$	Yah oto beema ke ubharate huye baajaar mein parvesh karane kaa paryaas kar rahee thee, jismein chaalakon ko apni milij ke adhaar par bhugtaan karnaa tha
Moses	उभरते बाजार में टूटा ऑटो बीमा, जहां चालक माइलेज का भुगतान करेंगे
Moses$_T$	Ubharate baajaar mein tootaa oto beemaa, jahaan chaalak milij kaa bhugtaan krenge
RNN Search	यह ऑटो बीमा में एक उभरते बाजार में टूटना चाह रहा था, जिसमें ड्राइवर अपने लाभ के आधार पर भुगतान करेंगे
RNN Search$_T$	Yah oto beemaa mein ek ubharate bahaar mein tutnaa chaah rhaa thaa, jismein draivar apne lab ke adhaar par bhugtaan krenge
Transformer	यह ऑटो बीमा में एक उभरते बाजार में टूटना चाह रहा था, जहां चालक माइलेज का भुगतान करना था
Transformer$_T$	Yah oto beemaa mein ek ubharate bahaar mein tutnaa chaah rhaa thaa, jahaan chaalak milij kaa bhugtaan karnaa tha
CE-Encoder	यह ऑटो बीमा के उभरते बाजार में प्रवेश करने का प्रयास कर रही थी, जहां ड्राइवर अपनी माइलेज के आधार पर भुगतान करेंगे
CE-Encoder$_T$	Yah oto beemaa ke ubharate baajaar mein parvesh karne kaa paryaas kar rahee thee, jahaan draivar apni milij ke adhaar par bhugtaan krenge

Conclusion

The above findings draw a picture of machine translation on a context basis. Our approach can be tuned easier as compared to the existing methods. Through BLEU, NIST and METEOR evaluation system, we evaluated the performance of the proposed model of CE-Encoder, which is a context-based recurrent encoder for translation and compared the results of the evaluation systems. Our translated target Hindi sentences from the source English sentences have found the proposed model superior in accuracy than other strong baseline models. Significantly, the comparison of the results of the BLEU system with the conventional translation system shows that our model achieved scores of 44.34 using the CFILT dataset and 28.56 using the ILCC dataset, which is higher than the scores obtained by the other translation approaches. The results demonstrate that our model performed better than Moses and BiRNN models and achieved scores of 13 and 10 more points for CFILT, whereas 10 and 7 more points for ILCC using the scores of BLEU, respectively. In the future, we would like to apply our approach to other language pairs and for other datasets

of large sentence lengths. We will consider the BERT model and compare the performance with our CE-Encoder in the future.

Acknowledgment

This study was supported by Council of Scientific and Indian Research (CSIR) with file no. 09/263(1152)/2018-EMR-I. We are also thankful to Department of Science and Technology (DST) for support through PURSE Grant. We also acknowledge the support from C-DAC for providing ILCC data set.

Author's Contributions

Mani Bansal: Proposed the idea, carried out the experiments and writing the manuscript.

D. K. Lobiyal: Fine tuning of the idea, Corrections and shaping of the paper.

Ethics

The authors confirm that this article is original and contains unpublished work. Both authors have read and approved the manuscript and there are no ethical issues involved.

References

Agrawal, R., & Sharma, D. M. (2017, August). Experiments on Different Recurrent Neural Networks for English-Hindi Machine Translation. In CS & IT-CSCP (pp. 63-74). https://csitcp.net/paper/7/710csit06.pdf

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. https://arxiv.org/abs/1409.0473

Banik, D., Ekbal, A., & Bhattacharyya, P. (2020). Statistical machine translation based on weighted syntax–semantics. Sādhanā, 45(1), 1-12. https://link.springer.com/article/10.1007/s12046-020-01427-w

Bhatnagar, S., & Chatterjee, N. (2020). Neural machine translation of Hindi and English. Journal of Intelligent and Fuzzy Systems, (Preprint), 1-9. https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs179873

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555. https://arxiv.org/abs/1412.3555

Costa-jussà, M. R., Allauzen, A., Barrault, L., Cho, K., & Schwenk, H. (2017). Introduction to the special issue on deep learning approaches for machine translation. Computer Speech and Language, *46*, 367-373. https://www.sciencedirect.com/science/article/abs/pii/S0885230816303965

Forcada, M. L., (2017). Making sense of neural machine translation. Translation spaces, 6(2), 291-309. https://www.jbe-platform.com/content/journals/10.1075/ts.6.2.06for

Gaikwad, N. (2020). Comparison of Word2vec with Hash2vec for Machine Translation. https://scholarworks.sjsu.edu/etd_projects/919/

Grundkiewicz, R., & Heafield, K. (2018, July). Neural machine translation techniques for named entity transliteration. In Proceedings of the Seventh Named Entities Workshop (pp. 89- 94. https://www.aclweb.org/anthology/W18-2413.pdf

Gupta, P., Costa-jussà, M., Rosso, P., & Banchs, R., (2016). A deep source-context feature for lexical selection in statistical machine translation. Pattern Recognition Letters 75, 24-29. doi.org/10.1016/j.patrec.2016.02.014

Gupta, S., He, P., Meister, C., & Su, Z., (2020). Machine translation testing via pathological invariance. 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 863-875. https://dl.acm.org/doi/abs/10.1145/3368089.3409756

Guzmán, F., Joty, S., Màrquez, L., & Nakov, P., (2017). Machine translation evaluation with neural networks. Computer Speech & Language, 45, 180-200. https://www.sciencedirect.com/science/article/abs/pii/S0885230816301693

Jang, M., Seo, S., & Kang, P., (2019). Recurrent Neural Network-Based Semantic Variational Autoencoder for Sequence-to-Sequence Learning. Information Sciences. doi.org/10.1016/j.ins.2019.03.066

Jaya, K., & Gupta, D., (2016). Exploration of Corpus Augmentation Approach for English-Hindi Bidirectional Statistical Machine Translation System. International Journal of Electrical and Computer Engineering (IJECE), Vol.6, No.3, 1059-1071. doi.org/10.11591/ijece.v6i3.8904

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. V. D., Graves, A., & Kavukcuoglu, K. (2016). Neural machine translation in linear time. arXiv preprint arXiv:1610.10099. https://arxiv.org/abs/1610.10099df

Kingma, D. P., & Ba, J., (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. https://arxiv.org/abs/1412.6980

Knowles, R., & Koehn, P., (2018). Context and Copying in Neural Machine Translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 3034-3041. https://aclanthology.org/D18-1339.pdf

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., & Bertoldi, N. (2007). Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, 177-180. https://aclanthology.org/P07-2045.pdf

Kunchukuttan, A., Khapra, M., Singh, G., & Bhattacharyya, P., (2018). Leveraging Orthographic Similarity for Multilingual Neural Transliteration. Transactions of the Association for Computational Linguistics, 6, 303-316. https://direct.mit.edu/tacl/article-abstract/doi/10.1162/tacl_a_00022/43438

Lample, G., Ott, M., Conneau, A., Denoyer, L., & Ranzato, M. A. (2018). Phrase-based & neural unsupervised machine translation. arXiv preprint arXiv:1804.07755. https://arxiv.org/abs/1804.07755/S0167865516000738

Liu, Y., Wang, K., Zong, C., & Su, K. Y. (2019). A unified framework and models for integrating translation memory into phrase-based statistical machine translation. Computer Speech & Language, 54, 176-206. https://www.sciencedirect.com/science/article/abs/pii/S0885230817302590

Ma, X., Zhou, C., Li, X., Neubig, G., & Hovy, E. (2019). Flowseq: Non-autoregressive conditional sequence generation with generative flow. arXiv preprint arXiv:1909. 02480. https://arxiv.org/abs/1909.02480

Narayan, R., Chakraverty, S., & Singh, V. P. (2016). Quantum neural network based machine translator for English to Hindi. Applied Soft Computing, 38, 1060-1075. doi.org/10.1016/j.asoc.2015.08.031

Ojha, A. K., Chowdhury, K. D., Liu, C. H., & Saxena, K. (2018). The RGNLP machine translation systems for WAT 2018. arXiv preprint arXiv:1812.00798. https://arxiv.org/abs/1812.00798

Parida, S., & Bojar, O. (2018). Translating short segments with nmt: A case study in English-to-Hindi. https://rua.ua.es/dspace/bitstream/10045/76083/1/EAMT2018-Proceedings_25.pdf http://rua.ua.es/dspace/handle/10045/76083

Pathak, A., & Pakray, P. (2019). Neural machine translation for indian languages. Journal of Intelligent Systems, 28(3), 465-477. https://www.degruyter.com/document/doi/10.1515/jisys-2018-0065/html

Rahimi, R., Shakery, A., & King, I. (2016). Extracting translations from comparable corpora for Cross-Language Information Retrieval using the language modeling framework. Information Processing and Management, 52(2), 299-318. doi.org/10.1016/j.ipm.2015.08.001

Ratnam, D. J., Soman, K. P., Bijimol, T. K., Priya, M. G., & Premjith, B. (2021). Hybrid Machine Translation System for the Translation of Simple English Prepositions and Periphrastic Causative Constructions from English to Hindi. In Applications in Ubiquitous Computing (pp. 247-263). Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-030-35280-6_12

Saini, S., & Sahula, V. (2018, March). Neural machine translation for English to Hindi. In 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP) (pp. 1-6). IEEE. https://ieeexplore.ieee.org/abstract/document/8464781/

Saini, S., & Sahula, V. (2021). A novel model based on Sequential Adaptive Memory for English–Hindi Translation. Cognitive Computation and Systems, 3(2), 142-153. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ccs2.12011

Sen, S., Banik, D., Ekbal, A., & Bhattacharyya, P. (2016, December). Iitp English-Hindi machine translation system at wat 2016. In Proceedings of the 3rd Workshop on Asian Translation (WAT2016) (pp. 216-222). https://www.aclweb.org/anthology/W16-4622.pdf

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with sub word units. arXiv preprint arXiv:1508.07909. https://arxiv.org/abs/1508.07909

Sharma, H., & Singh, H. (2021). Analysis of the Effect of Topic Modeling on General Corpus Mixed with In-Domain Text for English-Hindi Translation. In Rising Threats in Expert Applications and Solutions (pp. 777-783). Springer, Singapore. doi.org/10.1007/978-981-15-6014-9_93

Singh, S. P., Kumar, A., Darbari, H., Tailor, N., Rathi, S., & Joshi, N. (2017, March). Intelligent English to Hindi Language Model Using Translation Memory. In International Conference on Information and Communication Technology for Intelligent Systems (pp. 487-496). Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-319-63645-0_55

So, D., Le, Q., & Liang, C. (2019, May). The evolved transformer. In International Conference on Machine Learning (pp. 5877-5886). PMLR. http://proceedings.mlr.press/v97/so19a.html

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. Seventh international conference on spoken language processing. https://www.isca-speech.org/archive/icslp_2002/i02_0901.html

Sundermeyer, M., Alkhouli, T., Wuebker, J., & Ney, H. (2014, October). Translation modeling with bidirectional recurrent neural networks. In Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 14-25). https://aclanthology.org/D14-1003.pdf

Vaswani, A., Shazeer, N., & Parmar, N. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Weissenborn, D., Kiela, D., Weston, J., & Cho, K. (2018). Contextualized Role Interaction for Neural Machine Translation. https://openreview.net/forum?id=ryx3_iAcY7

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. arXiv preprint arXiv:1212.5701. https://arxiv.org/abs/1212.5701

Zhou, J., Cao, Y., Wang, X., Li, P., & Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. Transactions of the Association for Computational Linguistics, 4, 371-383. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00105/43368/Deep-Recurrent-Models-with-Fast-Forward