BUILDING AN EFFECTIVE MT SYSTEM FOR ENGLISH-HINDI USING RNN'S

Ruchit Agrawal[1] and Dipti Misra Sharma[2]

[1]Language Technologies Research Center, IIIT Hyderabad
[2]Head, Language Technologies Research Center, IIIT Hyderabad

ABSTRACT

Recurrent Neural Networks are a type of Artificial Neural Networks which are adept at dealing with problems which have a temporal aspect to them. These networks exhibit dynamic properties due to their recurrent connections. Most of the advances in deep learning employ some form of Recurrent Neural Networks for their model architecture. RNN's have proven to be an effective technique in applications like computer vision and natural language processing. In this paper, we demonstrate the effectiveness of RNNs for the task of English to Hindi Machine Translation. We perform experiments using different neural network architectures - employing Gated Recurrent Units, Long Short Term Memory Units and Attention Mechanism and report the results for each architecture. Our results show a substantial increase in translation quality over Rule-Based and Statistical Machine Translation approaches.

KEYWORDS

Machine Translation, Recurrent Neural Networks, LSTMs, GRUs, English-Hindi MT.

1.INTRODUCTION

Deep learning is a rapidly advancing approach to machine learning and has shown promising performance when applied to a variety of tasks like image recognition, speech processing, natural language processing, cognitive modelling and so on. Deep Learning involves using large neural networks for training a model for a specific task. This paper demonstrates the application of deep learning for Machine Translation of English *!* Hindi, two linguistically distant and widely spoken languages. The application of deep neural networks to Machine Translation has been demonstrated by (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014) and it has shown promising results for various language pairs.

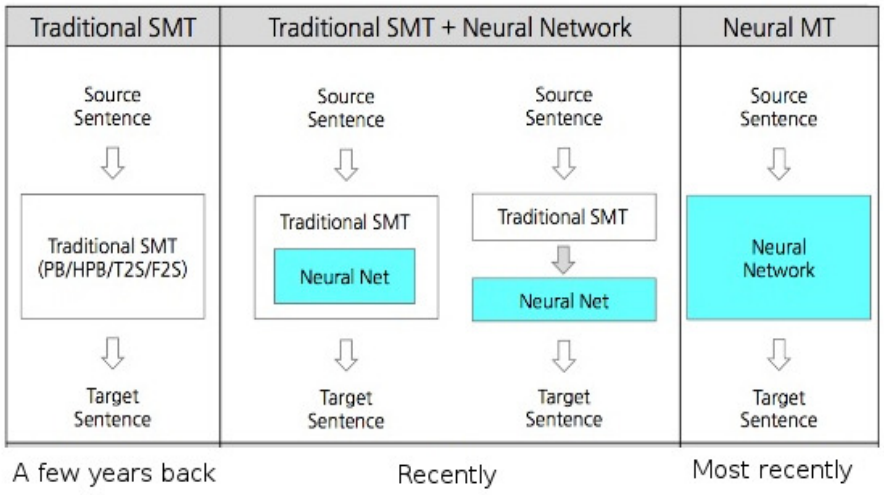
In this paper, we experiment with different deep learning architectures. These include Gated Recurrent Units (GRUs), Long Short Term Memory Units (LSTMs) and addition of attention mechanism to each of these architectures. We demonstrate that the best performance for English -> Hindi MT is generally obtained using Bi-directional LSTMs with attention mechanism and in some cases with GRUs with attention mechanism. The Bi-directional LSTMs generally show better performance for compound sentences and larger context windows.

We show manual samples of output translations and provide their evaluation to demonstrate the effectiveness of different architectures.
We describe the motivation behind the choice of RNNs in detail in Section 3. We briefly discuss related work in Section 2, followed by the description of our neural network model in Section 4. The experiments and results are discussed in Section 5. The paper is concluded in Section 6.

DOI : 10.5121/ijaia.2017.8504

2. RELATED WORK

The usage of large neural networks for Natural Language Processing (NLP) tasks was initially proposed by (LeCun et al., 2015) in his feed-forward neural language model. The neural Language Model he proposed is very similar to the current existing Language Models.



The input n-gram is projected into an embedding space for each word and passes to big output layer.

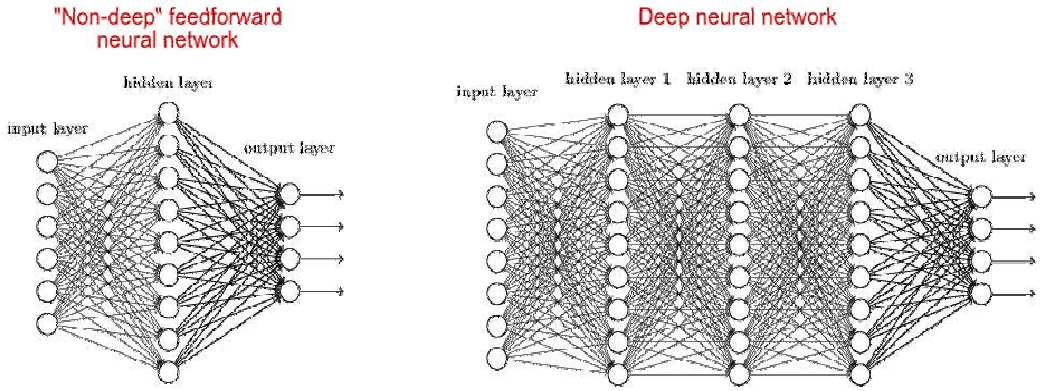


Figure 1: A comparison of feedforward neural networks with Recurrent Neural Networks

This novel idea was then used by several researchers who tried to integrate it with Machine Translation systems ((Auli et al., 2013) and (Cho et al., 2014)).

(Sutskever et al., 2014) was a breakthrough for Machine Translation, introducing the ”seq2seq” (Sequence to sequence) model which was the first model based completely on neural networks and achieving accuracy comparable to the State-of-the-Art SMT systems. They proposed the usage of a Recurrent Neural Network model with the encoders and decoders comprising of LSTMs or GRUs. They propose running the encoder over the source sentence, producing a hidden state and then running another RNN (decoder) to generate the output one word at a time.

The bottleneck to this approach was that the entire translation is a fixed sized vector. There have been different techniques (like padding) to rectify this issue.

Anusaaraka (Bharati et al., 1994) is an English to Hindi Machine Translation, primarily Rule-based, but employing a parser which uses statistical approaches (De Marneffe et al., 2006).

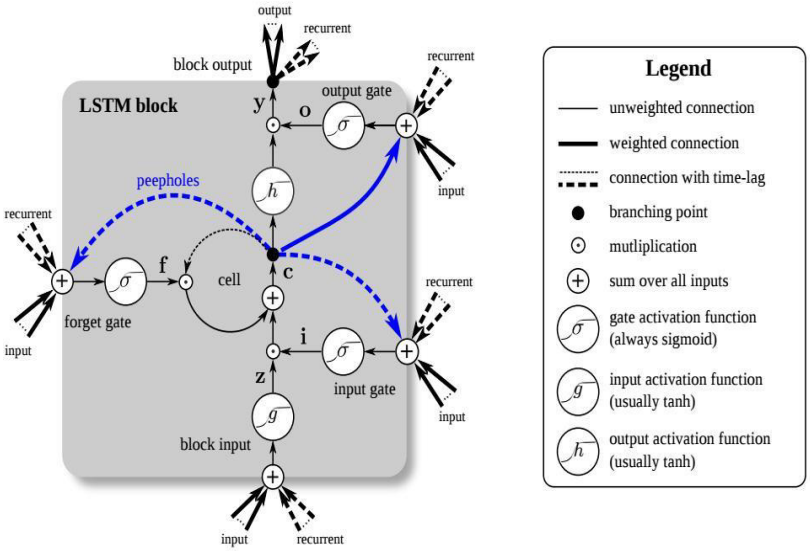


Figure 2: Structure of an LSTM unit

3. MOTIVATION BEHIND USING RECURRENT NEURAL NETWORKS

Traditional Neural Networks have a huge RAM requirement and are not quite feasible in their best settings where they achieve their highest accuracies. Additionally, they are not designed to deal with sequential information. We explain this below :

One important property of machine translation, or any task based on natural languages, is that we deal with variable-length input and output. For example; if the input X=(x_1; x_2; : : : ; x_T) and output Y=(y_1; y_2; : : : ; $y_{T'}$); The lengths of the sequences i.e. T and T' are not fixed.

On the other hand, one of the major assumptions in feedforward neural networks is the idea of fixed length, i.e. the size of the input layer is fixed to the length of the input sequence. The other major assumption is the idea of independence - that different training examples (like images) are independent of each other. However, we know of temporal sequences such as sentences or speech, there are short and long temporal dependencies that have to be accounted for.

To deal with these types of variable-length input and output, we need to use a recurrent neural network (RNN). Widely used feed-forward neural networks, such as convolutional neural networks, do not maintain internal state other than the network's own parameters. Whenever a single sample is fed into a feed-forward neural network, the network's internal state, or the activations of the hidden units, is computed from scratch and is not influenced by the state computed from the previous sample. On the other hand, an RNN maintains its internal state while

reading a sequence of inputs, which in our case will be a sequence of words, thereby being able to process an input of any length.

Recurrent Neural Networks (RNNs) also address the independence issue - they facilitate the preservation as well as processing of information that has a temporal aspect involved. For example; a sequence of words has an order, and hence a time element inherent in it. A model which takes this into consideration is needed for efficient performance. This is not possible if we employ feed-forward neural networks. Thus, Recurrent Neural Networks can not only learn the local and long term temporal dependencies in the data, but can also accommodate input sequences of variable length.

The RNN's thus help in converting the input sequence to a fixed size feature vector that encodes primarily the information which is crucial for translation from the input sentence, and ignores the irrelevant information. Figure 1 shows a comparison of feed-forward neural networks with recurrent neural networks.

Long Short Term Memory (LSTM) units are a type of RNNs which are very good at preserving information through time-steps over a period of time. Figure 2 shows the structure of an LSTM unit. One key advance in LSTMs in recent years has been the concept of bi-directional encoder and decoder framework. When we employ bidirectional LSTMs, we end up with two hidden states - one in the forward direction and one in the backward direction. This allows the network to learn from the text. Often, even more than two layers are used. Thus there will be multiple layers stacked on top of each other - this is generally only in huge training data conditions. Each one of these has a set of weights inside it, and learns and affects the one above it. The final state represents everything that is in the source words. Bi-directional LSTMs generally work the best specially when complemented with the attention mechanism.

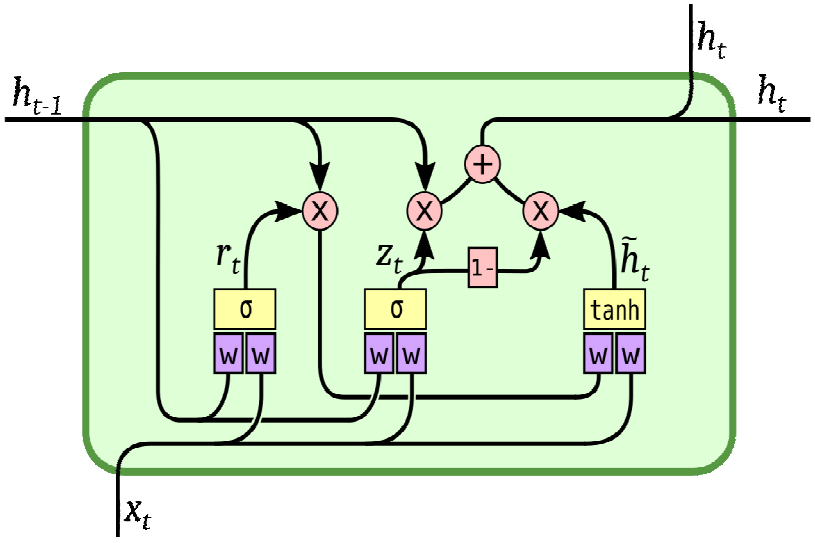


Figure 3 Structure of Gated Recurrent Unit

LSTM

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

GRU

$$
\begin{aligned}
z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\
r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \\
\tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

Figure 4 Mathematical formulation of LSTM and GRU

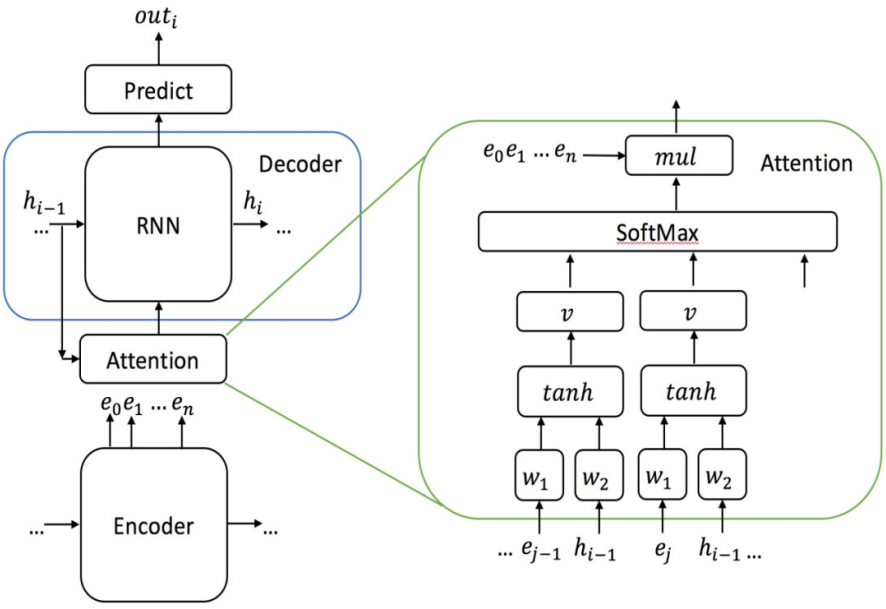


Figure 5 The attention mechanism

After the encoding process, we are left with a context vector - which is like a snapshot of the entire source sequence and is used further to predict the output. We have a dense layer with softmax similar to a feed-forward neural network, but the difference is that it is time distributed i.e. we have one of these for each time step. The top layer thus has one neuron for every single word in the vocabulary.

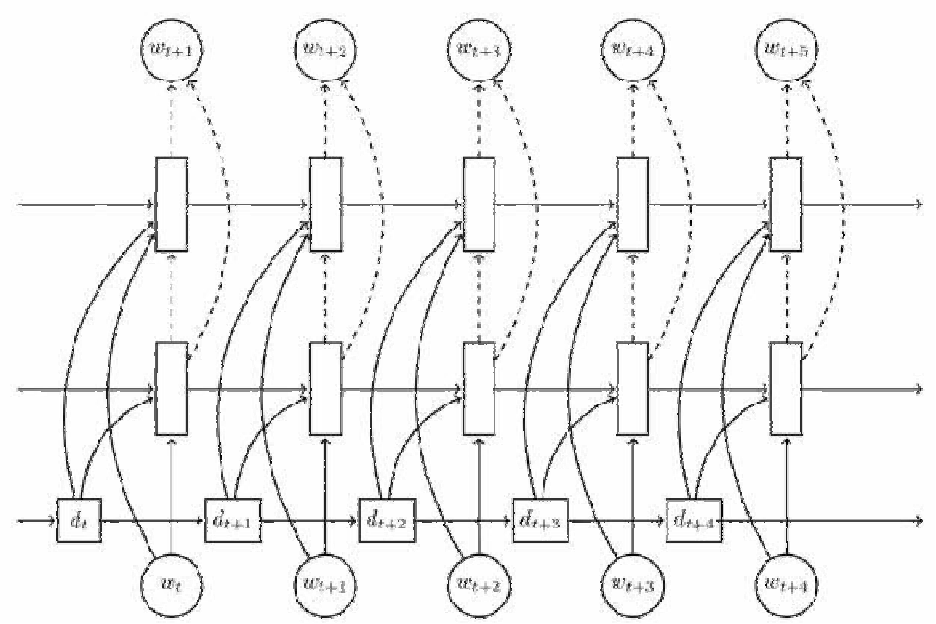


Figure 6: A two-layered LSTM architecture which we employ in our experiments

4. FORMULATION OF OUR MODEL

In order to train the recurrent neural networks, we take the cost function and obtain its derivative with respect to the weight in question. We then move this derivative through the nested layer of computations using the chain rule.

In other words, the output of the previous layer is multiplied by the weight matrix and added to a bias and then passed on to an activation function.

$$y_k = g(W \mathbf{y}_{k\ 1} + \mathbf{b}) \qquad (1)$$

Table 1: Different Hindi translations corresponding to the English sentence - “Shyam has given the book to Manish.” (Due to word order)

	Hindi	Transliteration
Sent : 1	□□□□ □□ □□□□□ □□ □□□□□ □□□□ ।	manIRa ko SyAma ne kiwAba xe xI
Sent : 2	□□□□□ □□ □□□□ □□ □□□□□ □□□□ ।	SyAma ne manIRa ko kiwAba xe xI

Table 2: Anusaaraka scores on ILCI test data

BLEU	NIST	METEOR	RIBES
6.98	3.68	0.164	0.592

We use a recurrent connection convert the linear unit of feed-forward neural network to a recurrent unit so that now the activity of the unit h_t not only depends on x_t (the input) multiplied by the weight matrix, but also on its activity at the previous timestep. The following equation shows this phenomenon :

$$h^{(t)} = g_h(W_1 x^{(t)} + W_R h^{(t\ 1)} + b_h) \qquad (2)$$

Table 3: Two different translations corresponding to the same English sentence - from ILCI test data (Many-to-many mapping between vocabulary)

T est	□□□□ □□□□□□□□□ □□□□□□□□ □□□□ □□□□ □□□□□□□□□□ □□ □□□□□□□□□□ ह
T l	wAjA sAzseM Ora camacamAwe xAzwa Apake vyakwiwva ko niKArawe hEM
T s	Fresh breath and shining teeth enhance your personality .
LST M	□□□□ □□□□ और □□□□□□□ □□□□ □□□□ □□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□
T s	wAjI sAzsa Ora camakaxAra xAzwa Apake vyakwiwva meM cAra cAzxa lagAwe hEM
T l	Fresh breath and shining teeth enhance your personality.

T l : Transliteration, T s : Translation, LST M : Long Short Term Memory, T est : Sentence from ILCI Test data

The second term $W_R h^{(t\ 1)}$ depends on the activity at the previous timestep multiplied by a recurrent weight matrix. We also want to be able to retrieve an output from this unit and this is done by adding a linear operation as described in the following equation :

$$y^{(t)} = g_y(W_y h^{(t)} + b_y) \qquad (3)$$

Here, $y^{(t)}$ is a function of $h^{(t)}$ multiplied by weight matrix w and passed through a non-linear activation function. This is the basic element of the recurrent neuron which we use in our RNN architectures.

The process can be visualized as the input sequence being compressed by the RNN into an intermediate representation in the form of a fixed dimensional vector. So, if the vector $h_{t\ 1}$ describes the history of the sequence at timestep t, the new internal state (the updated vector) h_t will be computed by the network, effectively compressing the preceding symbols ($x_1; x_2; : : : ; x_{t\ 1}$) as well as the new symbol x_t. The following equation shows this :

$$h_t = \phi\ (x_t, h_{t\ 1})$$

Here, ϕ is a function which takes the new information unit x_t and the hidden state $h_{t\ 1}$ as input. (h_0 can be assumed to be a vector containing zeroes).

4.1 LSTM and GRU cells

The LSTM cells are an improvement over vanilla RNN units. The structure of an LSTM cell is shown in Figure 2. In addition to introduction of hidden layers, there is a gating mechanism consisting of an input gate, forget gate and an output gate. We have an inner memory, which is linear to time backwards. The forget and the input gate control what we want to keep from the past as an information and what we want to accept new from the input. (Bahdanau et al, 2014) use a Gated Recurrent Unit, whose structure is shown in Figure 3.

The equations for the formulation of LSTM as well as GRU are described in Figure 4. Here, W, W_r and W_u are the input weight matrices; U, U_r and U_u are the recurrent weight matrices and b, b_r and b_u are the bias vectors. The main difference between a GRU and LSTM is how the u_t works. So rather than having a forget gate and an input gate that are both different to get the information inside the inner state, we have the u t which takes the past hidden value and we have (1 - u_t) which lets the entering of new information. Since u_t is vectorial, it is a combination of many neurons, and the computation is neuron wise, unlike a typical dot product.

4.2 Encoders and Decoders

We have many time-steps for an encoder. The input vector x is encoded into a state ”thought vector” S_i as shown in the Figure 6. Decoders, in turn, decode the state, ”thought vector” through time, thereby generating a sequence-to-sequence architecture when pipelined with the encoder. The thought vector which we use in our experiments is of the of the dimension of 1000. One bottleneck using this approach is the handling of long sequences. The backpropagation offset through time with the backward link makes the neural network more resilient to learning, correcting its errors as it goes forward. One common technique to improve accuracy is the reverse the input sequence and feed the reversed sequence to the encoder. This makes the related words closer to each other in the encoder and the decoder.

4.3 Lookup Tables and Embeddings

We pass on the numbers as input to the neural network rather than plain words.This also helps the model to learn related concepts easily. Lookup tables are used for this. Words that occur with very low frequency are discarded and replace with an <UNK> (UNK stands for Unknown) token. This is done to reduce the size of the vocabulary, since the computational power and complexity increases linearly with increase in the vocabulary size. For example, if there we have a vocabulary size of 200000, there has to be a dense layer of 200000 neurons at the top, which goes into 28a softmax to predict the word output. To avoid this, we use lookup tables and word embeddings. Word embeddings allow us to extract more semantic information from the words.
Often, pre-trained embeddings like word2vec or GlOVe are used. Since the embeddings are generally trained across billions of words, they are able to spot relationships and leverage semantic information in the neural network.

4.4 Padding

A sequence to sequence model generally has a fixed length for the sequence, for example 30 time steps. To achieve a common length across input sequences, we use padding. We pad the sequence with special tokens to achieve the desirable length.All the input sequences must have a common length and all output sequences must have a common length, however the input sequence length

need not be the same as the output sequence length.The padding tokens employed by us belong to the following categories (make a table): <PAD> denotes a padded zero input , <EOS> denotes end of sentence, <GO> tells the decoder to start functioning, <OOV> represents an out-of-vocabulary token, <UNK> represents an unknown token and <ES2> specifies the target language. These tokens thus can be thought of as giving conditional information to the network as to what it should be doing.

4.5 Attention mechanism

The traditional encoder-decoder framework at first encodes the source sentence into a single vector representation which is used by the decoder to predict every single word in the output. In other words, each input word is used equally used for translating the sentence, and all words are used by the decoder at each time step.With the attention mechanism, the decoder computes a set of attention weights which is applied at the input sequence at each timestep. The set of attention weights changes over time. A weighted sum using the attention weights of the input generates each word. Since the attention weights change with time, the model can ”focus” in different places as the translation process moves forward. The attention model keeps track of the source hidden states as a memory pool. The reference is then done to the words according to the weights assigned to them.Given the context vector and the target hidden state history, the attentional vector ($\tilde{h}_t$) can be computed as follows:

$$\tilde{h}_t = \tanh(W_c [c_t ; h_t])$$

We also employ Input-feeding, an effective extension to attention mechanism where the ttentional vectors are fed to the next timesteps as shown in Figure 5.We try to predict the aligned position p_t , which defines a focused attention. This implies that we look only at a context window of (p_t - D, p_t + D) rather than looking at entire source hidden state.We perform experiments with different RNN architectures and the results are discussed in the next section. We select the optimal architecture after this experimentation. The final resultant architecture of our model is shown in Figure 6.

5.EXPERIMENTS AND RESULTS

We employ a sequence-to-sequence model with Recurrent Neural Networks to train our models. We conduct experiments on two and four layers of encoder and decoder respectively. We use the architecture as described in Section 4. We use the seq2seq model available in Tensorflow[1] to implement the above mentioned architecture.

For training the model, we extract 200,000 sentences from the HindEnCorp (Bojar et al., 2014) corpus. We employed pruning using appropriate rules to remove unsuitable sentences. For example, all sentences of length greater than fifty were removed from the corpus. The reason was low scalability of neural networks to translate sentences of length greater than 50. Also, sentences of length less than three were removed to discourage memorization, instead of syntactic and semantic learning of concepts. Pruning was also done to remove special characters and hyperlinks from the sentences.

After removing discrepancies, rest of the sentences were randomly shuffled to create the parallel training corpus. We test the performance of our model using the ILCI test set (Jha, 2010) and the WMT 2014 English-Hindi test set.

Table 4: Results - Comparison of metric scores obtained on two-layered and four-layered model at different stages

	Two Layers				Four Layers			
	BLEU	NIST	METEOR	RIBES	BLEU	NIST	METEOR	RIBES
Anusaaraka	6.61	2.51	0.156	0.592	6.72	3.33	0.158	0.567
GRU	15.19	4.45	0.149	0.727	16.41	2.82	0.149	0.64
LST M	15.32	4.42	0.21	0.74	16.85	4.48	0.16	0.69
BiLST M	15.39	4.31	0.228	0.73	17.31	4.62	0.23	0.763
GRU_{Att}	16.06	5.37	0.239	0.758	17.45	5.21	0.244	0.775
$LST\ M_{Att}$	16.76	5.43	0.246	0.760	17.63	5.49	0.251	0.788
$BiLST\ M_{Att}$	17.91	5.47	0.251	0.78	18.41	5.57	0.274	0.81

GRU : Gated Recurrent Unit
Anusaaraka :Rule-based Machine Translation for English-Hindi
LST M : Long Short Term Memory
Att : Attention Mechanism
BiLST M : Bi-directional LSTM
We use similar nomenclature in all tables hereby.

Table 5: Evaluating output quality : Different RNN architectures

T est	□□□□ □□□□□ □□□ □□□□□□□ □□□□□□
T l	isakA upacAra saBI aspawAloM meM hE
T s	Its treatment is available in all hospitals.
Anusaaraka	□□□□ लए अब □□□□□□ □□ एक □□□□ □□□
T l	isake lie aba upalabXa BI eka golI hE
T s	For this, there is now available also a pill.
GRU	□□□□ □□□□□ □□□ □□□ □□□□□□□ □□□□□□□□□ □□□
T l	isakA nixAna saBI saBI aspawAloM meM upalabXa hE
T s	The solution for this is available in all all hospitals.
LST M	□□□□ □□□□ □□□ □□□□□□□ □□□□□□□□□ □□□
T s	usakA ilAja saBI aspawAloM meM upalabXa hE
T l	The treatment for that is available in all hospitals.
BiLST M	□□□□ □□□□□ □□□ □□□□□□□ □□□□□□□□□ □□□
T l	isakA upacAra saBI aspawAloM meM upalabXa hE
T s	The treatment for this is available in all hospitals.

Comparing the performance of different neural network architectures (without attention mechanism)

Table 6: Evaluating output quality : Adding Attention Mechanism

GRU_{Att}	□□□□ □□□ क □□□□□□□□□□□□□□□□□□ □□ ज र □□□□□ □□□□□
T l	apanI roja kI xinacaryA meM vyAyAma ko jarUra SAmila kareM
T s	Do include exercise in your daily routine.
$LST\ M_{Att}$	□□□□□□□□□□ □□ □□□□ □□□□□ □□□□□□□□□□□□□□□□ □□□□□
T l	eksarasAija (transliteration) ko apane xEnika xinacaryA meM SAmila kareM
T s	Include exercise in your everyday routine.
$BiLST\ M_{Att}$	□□□□□□□ □□ □□□□ □□□□□ □□□□□□□□□□□□□□□ □□□□□
T l	vyAyAma (translation) ko apanI xEnika xinacaryA meM SAmila kareM
T s	Include exercise in your everyday routine.

We observe that our model is able to produce grammatically fluent translations, as opposed to traditional approaches. Some problems which still need to be solved are presence of repeated tokens and unknown or unmapped words. A bi-directional LSTM model with attention mechanism shows improvement over normal RNN's in both these aspects.

Table 7: Evaluating output quality : Two layers vs. Four layers

T est	40 □□□ □□ अ धक □□□□□ □□□ □□□□□□□ क □□□□□□□□□□□ □□□□□ क □□□□ □□□□□ ।
T l	40 sAla se aXika Ayu ke saBI vyakwiyoM kI vArRika jAzca avaSya kI jAnI cAhie
T s	An annual check-up of everybody above the age of 40 years must be done .
$LST\ M_{2l}$	40 वष क □□□□ □□ ऊपर □□□□□□□□□□□ □□□□□□□□ □□□□□□□ क □□□□ □□□□ ।
T l	40 varRoM kI umra ke Upara prawyeka vyakwi kI vArRika jAzca karanI hogI
T s	An annual checkup of each person above 40 years of age will have to be done.
$LST\ M_{4l}$	40 वष क □□□□ ऊपर □□□□□□□□ □□□□□□ □□ □□□□□□□□□□□ □□ □□□□ □□□□ □□□□□ ।
T l	40 varRoM kI umra ke Upara prawyeka vyakwi kA vArRika jAzca kiyA jAnA cAhie
T s	An annual check-up of each person above 40 years of age should be done.
$BiLST\ M_{2l}$	40 □□□□□ □□□□ □□ अ धक □□□□□□□□ □□□□□□ क □□□□□□□□□□□ □□ □□□□ प □□ क □□□□ □□□□□ ।
T l	40 varRa kI umra ke Upara hara sAla vArRika jAzca karanI cAhie
T s	After 40 years of age, every year an annual checkup should be done.
$BiLST\ M_{4l}$	40 □□□□□ □□□□ □□ अ धक □□□□□□□□□□ □□□□□□□□ □□□□□□□ क □□ □□□□ □□□□□ ।
T l	40 varRa kI umra ke prawi hara eka vArRika testa karanI cAhie
T s	After 40 years of age, every one annual test should be done.

2l : Two layers, 4l : Four layers

Table 8: Results on WMT Test data

	BLEU	NIST	METEOR	RIBES
GRU	1.57	1.46	0.0738	0.277
Anusaaraka	4.40	2.72	0.12	0.488
LST M	6.57	2.89	0.163	0.611
BiLST M	8.42	3.26	0.198	0.67
BiLST M_{Att}	**9.23**	**3.67**	**0.211**	**0.71**

Performance evaluation on WMT test set

Table 4 demonstrates the performance of our model during various stages as measured by the above-mentioned metrics. We observe on manual inspection of samples that there is a significant improvement in performance over rule-based and statistical approaches by using deep neural networks, thereby producing quality translation as shown by the use of semantically correct synonyms. For example, Table 3 shows a sample sentence from the ILCI test corpus ($ILCI_{test}$) and its corresponding output obtained by our model. The English as well as Hindi meaning of both the sentences is the same, although they differ in their structure and words used in the Hindi output. The LSTM output displays an impressive usage of the phrase “cAra cAzxa lagAwe hEM” - a contextually suitable and semantically correct idiom in Hindi which conveys “enhancing of personality”.

Anusaaraka has a BLEU score of 6.98 on ILCI test data (Table 2). We observe a 4.72 point increase in the BLEU score by using *GRU*s. Similar improvements can be seen for other metrics by using different RNN architectures. Table 5 shows the variation in quality of translation obtained on using different RNN architectures. The Anusaaraka output does not make much sense (is syntactically as well as semantically poor) and the GRU a grammatically incorrect sentence. While the LSTM model produces a better translation with a minor error in pronoun usage, the Bidirectional LSTM model generates the correct output.

We demonstrate the effect of addition of attention mechanism in Table 6. Table 7 compares the output of two-layered model and four-layered model obtained on the different architectures using sample translations. We can observe that the four-layered model is able to perform better in many cases two-layered counterpart. The reason can be attributed to higher complexity of this model and sufficient data for training.

We also conduct experiments and report results on the WMT-14 corpus in Table 8. The results further improve on using Bi-directional LSTM with attention to give a BLEU score of 9.23, comparable to (Dungarwal et al., 2014), a system fully trained on the WMT training corpus.

6. Conclusion and Future Work

In this paper, we build sequence-to-sequence models using Recurrent Neural Networks. We experimented with Gated Recurrent Units, Long Short Term Memory Units and the attention mechanism. We demonstrated results using this approach on a linguistically distant language pair En *!* Hi and showed a substantial improvement in translation quality. We conclude that Recurrent Neural Networks perform well for the task of English-Hindi Machine Translation. The bidirectional LSTM units perform best, specially on compound sentences. Future work includes

performing experiments on other languages, especially among morphologically rich languages, like Indian to Indian language MT. We would like to explore MT for resource-scarce languages, in conditions where large parallel corpora for training are not available.

REFERENCES

[1] Gary Anthes. 2010. Automated translation of indian languages. Communications of the ACM 53(1):24– 26.

[2] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In EMNLP. volume 3, page 0.

[3] Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1994. Anusaraka or language accessor: A short introduction. Automatic Translation, Thiruvananthpuram, Int. school of Dravidian Linguistics .

[4] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. Natural language processing: a Paninian perspective. Prentice-Hall of India New Delhi.

[5] Roger M Blench and M Post. Rethinking sino-tibetan phylogeny from the perspective of north east indian languages. paper accepted for a volume of selected papers from the 16th himalayan languages symposium 2-5 september 2010 school of oriental and african studies, london. ed. Nathan Hill. Mouton de Gruyter .

[6] Ondrej Bojar, Vojtech Diatka, Pavel Rychlỳ, Pavel Stranák, Vít Suchomel, Ales Tamchyna, and Daniel Zeman. 2014. Hindencorp-hindi-english and hindi-only corpus for machine translation. In LREC. pages 3550–3555.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 .

[8] Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In ICML. pages 2067–2075.

[9] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of LREC. Genoa Italy, volume 6, pages 449–454.

[10] George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proceedings of the second international conference on Human Language Technology Research. Morgan Kaufmann Publishers Inc., pages 138–145.

[11] Piyush Dungarwal, Rajen Chatterjee, Abhijit Mishra, Anoop Kunchukuttan, Ritesh Shah, and Pushpak Bhattacharyya. 2014. The iit bombay hindi english translation system at wmt 2014. ACL 2014 page 90.

[12] Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3. Association for Computational Linguistics, pages 1152–1161.

[13] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pages 944–952.

[14] Girish Nath Jha. 2010. The tdil program and the indian langauge corpora intitiative (ilci). In LREC.

[15] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In EMNLP. 39, page 413.

[16] Nayan Jyoti Kalita and Baharul Islam. 2015. Bengali to assamese statistical machine translation using moses (corpus based). arXiv preprint arXiv:1504.01182 .

[17] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 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 . Association for Computational Linguistics, pages 177–180.

[18] Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. Machine translation 23(2):105–115.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. Nature 521(7553):436–444.

[20] Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In Proceedings of the International Workshop on Spoken Language Translation .

[21] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 .

[22] Anthony McEnery, Paul Baker, Rob Gaizauskas, and Hamish Cunningham. 2000. Emille: Building a corpus of south asian languages. VIVEK-BOMBAY- 13(3):22–28.

[23] 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 on association for computational linguistics. Association for Computational Linguistics, pages 311–318.

[24] Reinhard Rapp and Carlos Martin Vide. 2006. Example-based machine translation using a dictionary of word pairs. In Proceedings, LREC. pages 1268–1273.

[25] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming creating large training sets, quickly. In Advances in Neural Information Processing Systems. pages 3567–3575.

[26] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In Proceedings of association for machine translation in the Americas. volume 200.

[27] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In Interspeech. pages 194–197.

[28] 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.

[29] Nicola Ueffing, Gholamreza Haffari, Anoop Sarkar, et al. 2007. Transductive learning for statistical machine translation. In Annual Meeting-Association for Computational Linguistics. volume 45, page 25.

[30] Paul J Werbos. 1990a. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78:1550–1560.

[31] Paul J Werbos. 1990b. Backpropagation through time, what it does and how to do it. Proceedings of the IEEE 78.

[32] David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, pages 189–196.